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

Кафедра «Прикладная математика и информатика»
(наименование)

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

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

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Реализация криптографического алгоритма на эллиптических кривых

Обучающийся Р.М. Ерченко
(Инициалы Фамилия) (личная подпись)

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

Консультант к.п.н., доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Бакалаврская работа посвящена реализации криптографического алгоритма на эллиптических кривых.

Актуальность этой темы обусловлена развитием технологий и методов взлома, постоянно растет необходимость в более надежных криптографических методах. Эллиптические кривые представляют собой один из таких методов, который пока что не подвержен известным атакам, связанным с факторизацией или проблемой дискретного логарифма.

Целью бакалаврской работы является реализация криптографического алгоритма на эллиптических кривых.

Задачами бакалаврской работы являются описание теоретических основ эллиптических кривых, использующихся в криптографии, исследование алгоритмов криптографии на эллиптических кривых, реализация алгоритма обмена ключами ECDH и алгоритма цифровой подписи ECDSA.

Объектом исследования бакалаврской работы являются эллиптические кривые.

Предметом исследования бакалаврской работы являются алгоритмы на эллиптических кривых.

В первой главе описывается важные аспекты эллиптических кривых, математическая теория для дальнейшей реализации алгоритмов.

Во второй главе описана теория по трем основным направлениям: обмен ключами, цифровая подпись и шифрование.

В третьей главе показана программная реализация эллиптических кривых на языке python и реализация криптографических алгоритмов обмена ключами ECDH и цифровой подписи ECDSA.

Abstract

The title of the thesis is Implementing a cryptographic algorithm on elliptic curves.

The senior paper consists of an introduction, three parts, a conclusion, figures, a list of references, including foreign sources, and an appendix.

The key issue of the thesis is the implementation of cryptographic algorithms on elliptic curves. We address the problem of choosing a suitable elliptic curve cryptographic algorithm (ECC) that will provide the necessary level of security and efficiency.

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

The graduation work may be divided into several logically connected parts which are: description of important aspects of elliptic curves, mathematical theory for further implementation of algorithms; description of the theory in three main areas: key exchange, digital signature and encryption; the software implementation of elliptic curves in python and the implementation of cryptographic algorithms for key exchange ECDH and digital signature ECDSA.

Finally, we present the work on the successful implementation of two encryption algorithms on elliptic curves.

In conclusion we'd like to stress that asymmetric algorithms are needed to ensure a high level of security, performance and compatibility, which makes them indispensable in various fields, from Internet protocols and digital signatures to cryptocurrencies and IoT devices. Given the growing cyber threats and the advent of quantum computing, further research and development in this area remains critically important to ensure data and communications security.

Оглавление

Введение 5
Глава 1 Теоретические основы эллиптических кривых 7
1.1Математические операции над эллиптическими кривыми 7
1.1.1 Эллиптическая кривая. Сингулярность и несингулярность 7
1.1.2 Абелева группа 10
1.1.3 Арифметика эллиптических кривых 12
1.1.4 Умножение точки на скаляр 14
1.1.5 Задача логарифмирования на эллиптической кривой 17
1.2 Конечные поля при эллиптические кривых 18
1.2.1 Виды полей над кривыми 18
1.2.2 Поле ***F***p и эллиптические кривые над ним 19
1.2.3 Операция сложения точек над конечным полем 22
1.2.4 Порядок группы на эллиптической кривой 23
1.2.5 Подгруппы на эллиптической кривой 25
1.2.6 Задача дискретного логарифмирования на эллиптической кривой 26
Глава 2 Алгоритмы шифрования на эллиптических кривых 28
2.1 Алгоритм обмена ключами 28
2.2 Алгоритм цифровой подписи 30
2.3 Смешанные алгоритмы 32
Глава 3 Программная реализация алгоритмов шифрования 35
3.1 Выбор средств 35
3.2 Структура проекта 35
3.3 Реализация методов и операций на эллиптических кривых 36
3.4 Реализация алгоритма обмена ключами 40
3.5 Реализация цифровой подписи 41
Заключение 44
Список используемой литературы 45
Приложение А Листинг программы 48

Введение

Эллиптические кривые (EC) в криптографии предоставляют эффективные и безопасные методы для выполнения ключевых операций, таких как шифрование, подпись и аутентификация. В последние годы они стали основой многих криптографических протоколов благодаря своим преимуществам в сравнении с традиционными методами, такими как RSA и DSA.

Актуальность работы заключается в росте принятия ECC в качестве основы для различных криптографических стандартов и протоколов. Протоколы, такие как TLS (Transport Layer Security), используют ECC для обеспечения безопасного обмена данными в интернете. Организации, такие как NIST (Национальный институт стандартов и технологий), рекомендуют использование ECC в своих руководствах по безопасности. Это делает исследование и реализацию ECC важными для обеспечения соответствия современным стандартам безопасности, «также криптография применима к устройствам с ограниченными вычислительными ресурсами, такими как мобильные телефоны, смартфоны 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» [1].

Целью бакалаврской работы является реализация криптографического алгоритма на эллиптических кривых.

Задачи бакалаврской работы:

- описать теоретические основы эллиптических кривых, использующиеся в криптографии, включая методы и операции;
- исследовать три криптографических алгоритма на эллиптических кривых;

– реализовать алгоритм обмена ключами ECDH и алгоритм цифровой подписи ECDSA.

Предметом исследования бакалаврской работы являются алгоритмы на эллиптических кривых.

Работа содержит три главы.

В первой главе описываются формулы и методы на эллиптических кривых. Затрагивается тема конечных полей, действительных чисел и задача логарифмирования.

Во второй главе описывается теория по трем алгоритмам шифрования. ECDH – обмен ключами. ECDSA – цифровая подпись. ECIES – смешанные алгоритмы криптографии.

В третьей главе описывается реализация двух алгоритмов шифрования, таких как обмен ключами и цифровая подпись.

Глава 1 Теоретические основы эллиптических кривых

1.1 Математические операции над эллиптическими кривыми

1.1.1 Эллиптическая кривая. Сингулярность и несингулярность

«Эллиптическая кривая была открыта благодаря исследованиям Ньютона в области кубических кривых, его открытие привело к формулам сложения точек, которые задали основу для одного из методов асимметричного шифрования» [23].

Шифрование на эллиптических кривых отличается от стандартных видов наличием эллиптической кривой. Кривая определяется в конечном поле, она симметрична по оси x. При проведении через нее линии не симметричной по двум осям будут образованы три точки, обычно называемые P, Q, R. «Эллиптическая кривая задается уравнением Вейерштрасса» [15]. Функция эллиптической кривой по Вейерштрассу представлена в формуле (1)

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

«Значения a и b являются константами, определяющими форму эллиптической кривой. В зависимости от этих коэффициентов можно построить некоторые кривые» [1].

Однако существуют и другие формулы эллиптических кривых, всё зависит от характеристики поля K. Например, если K=3, то уравнение кривой обозначается формулой (2)

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

Сингулярность и несингулярность относятся к свойствам кривой, связанным с ее геометрическим строением.

Примеры гладких эллиптических кривых, зависящих от a и b, приведены на рисунке 1.

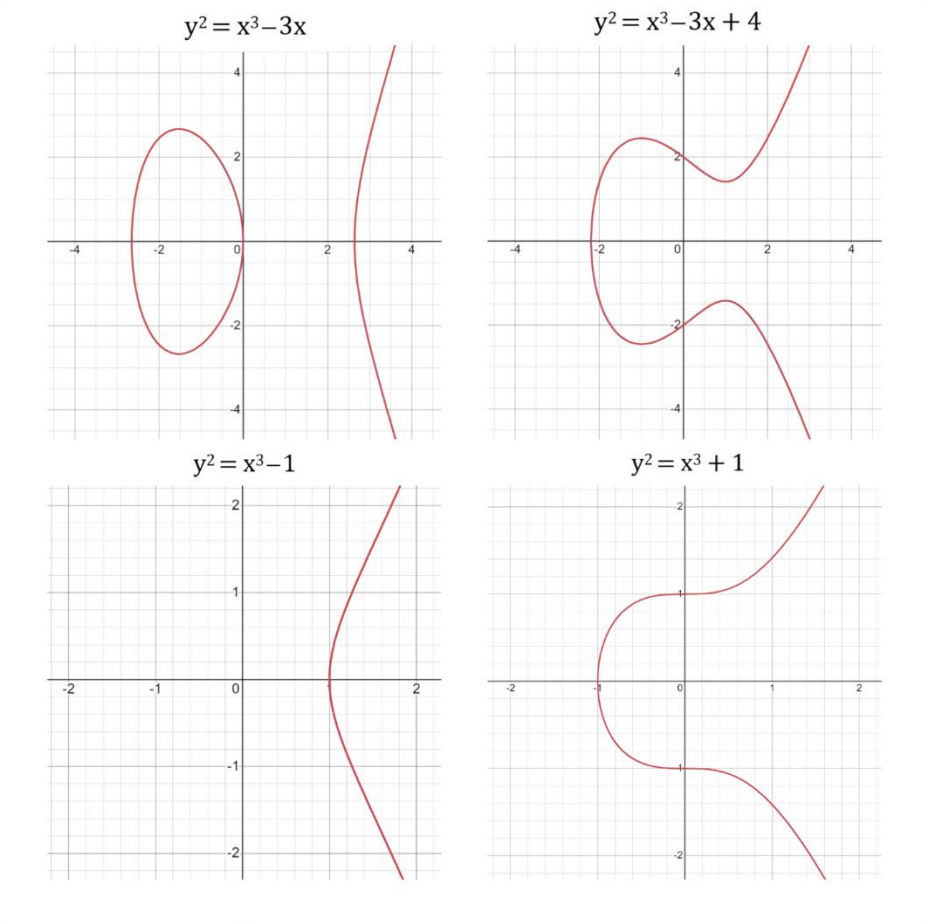

Рисунок 1 – Несингулярные кривые

Для криптографических систем эллиптическая кривая должна быть несингулярной, то есть она имеет три различных корня, в отличие от сингулярной кривой. Для того, чтобы определить несингулярность или сингулярность кривой можно вычислить дискриминант. По определению, нужно, «чтобы кривая не имела особых точек, самопересечений или точек возврата, тогда если дискриминант не равен нулю, то кривая будет считаться несингулярной» [25]. Дискриминант представлен в формуле (3)

$$4a^3+27b^2\neq 0. \quad (3)$$

Если кривая не имеет особых точек, тогда она делится на две части, на одной дискриминант будет положителен, а на другой отрицателен.

Сингулярные кривые не используются в криптографии, так как у них есть точки возврата или самопересечения. Такие кривые не являются эллиптическими в понимании математики, используемой в криптографии. Они не образуют абелеву группу во время сложения точек. Это является основным требованием для использования в криптографии. Сингулярная точка – это точка с координатами x и y, в которой частные производные функции f равны нулю. Пример для x представлен в формуле (4), формула для y аналогична за исключением замены в знаменателе.

$$\frac{\partial \mathrm{f}}{\partial x}(x, y) = 0. \quad (4)$$

Сингулярных кривых имеют особую узловую или каспидальную точку. В узловой точке кривая пересекает саму себя. Функция кривой с узловой точкой представлена в формуле (5).

$$y^2=x^2(x+1). \quad (5)$$

Каспидальная точка – это точка, в которой кривая имеет единственную касательную и выглядит как острый выступ. Функция кривой с каспидальной точкой представлена в формуле (6).

$$y^2=x^3. \quad (6)$$

Примеры сингулярных кривых с особыми точками соответственно ходу повествования представлены на рисунке 2.

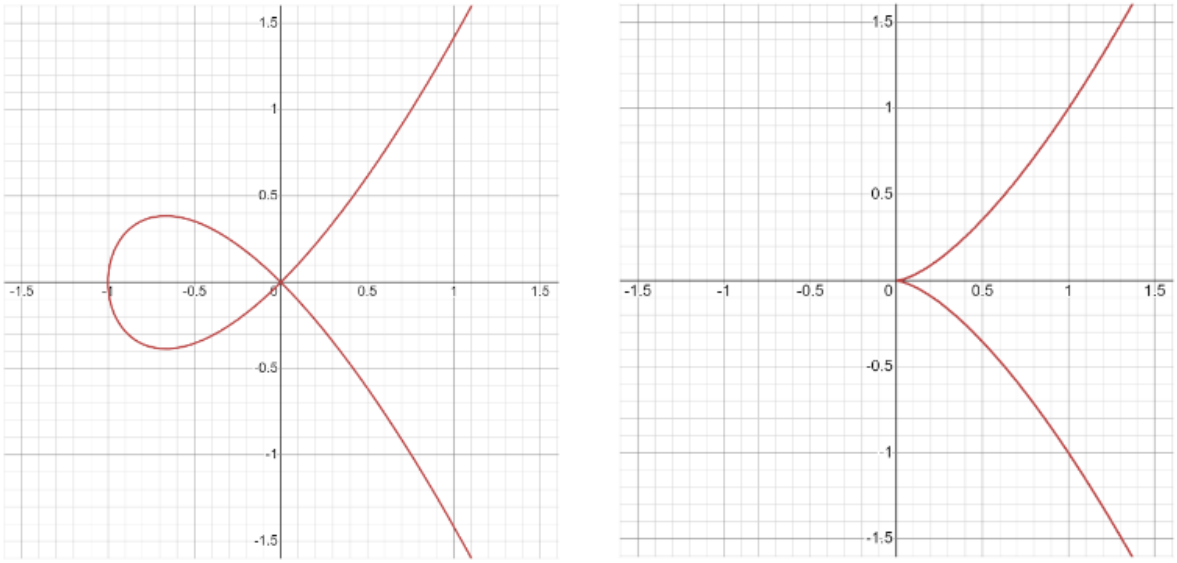

Рисунок 2 – Кривые с особыми точками

Узловые точки являются важным понятием в теории алгебраических кривых. Они представляют собой точки пересечения гладких ветвей кривой и обладают особыми алгебраическими и геометрическими свойствами, каспидальные точки имеют вид гладкой кривой, которая резко меняет направление в данной точке. Понимание узловых точек необходимо для углубленного изучения алгебраических кривых и их приложений в различных областях математики.

Таким образом, в криптографии используются именно несингулярные эллиптические кривые, которые обладают необходимыми математическими свойствами и обеспечивают достаточный уровень безопасности, так как сингулярные могут иметь неопределенную или слишком сложную для вычислений операцию сложения точек, а также не имеют четко определенной абелевой группы.

1.1.2 Абелева группа

Рассмотрим группу точек на эллиптической кривой. Здесь применяется операция, которая позволяет при сложении двух точек получить третью. Эти

точки вместе с операцией сложения образуют алгебраическую структуру – абелеву группу.

Приведем основные свойства абелевой группы:

- замкнутость в данном случае означает, что «при складывании точек P и Q на нашей кривой будет найдена новая третья точка R;
- ассоциативность здесь является простым примером равенства наших точек (P + Q) + R = P + (Q + R), результат сложения в данном случае не влияет на результат» [4];
- может существовать такой нейтральный элемент, если на эллиптической кривой существует особая точка, называемая точкой на бесконечности и обычно обозначаемая как O. Эта точка будет являться нулевым элементом и при сложении с первой точкой P результатом будет точка та же самая точка P;
- коммутативность в самом полном понимании являет собой операцию сложения двух точек P и Q нашей кривой. После этого можем получить равенство P + Q = Q + P. «Нетрудно сделать вывод, что здесь, как и в арифметике сложение двух элементов при их перестановке не имеет никакого значения» [4];
- «существование обратного элемента определяется тем, что у точки P есть обратная точка –P и в результате сложения этих двух точек получим точку на бесконечности O» [7].

Благодаря этим свойствам множество точек эллиптической кривой называется абелевой группой.

Эллиптические кривые являются примером одномерных абелевых многообразий.

«В математике абелева группа, также называемая коммутативной группой, представляет собой группу, в которой результат применения групповой операции к двум элементам группы не зависит от порядка, в котором они записаны. То есть групповая операция коммутативна. При

сложении в качестве операции целые и действительные числа образуют абелевы группы, и концепцию абелевой группы можно рассматривать как обобщение этих примеров» [5].

Группы точек необходимы в криптографии, так как используются для создания цифровых подписей, криптографических ключей и шифрования данных, имеют меньшие размеры ключей и обеспечивают высокий уровень безопасности по сравнению со стандартными асимметричными алгоритмами, такими как RSA.

1.1.3 Арифметика эллиптических кривых

Рассмотрим арифметику на кривой. «Сложение точек на эллиптической кривой является одной из основных операций, на которой основана криптография на эллиптических кривых. Эта операция позволяет определять сумму двух точек, лежащих на данной эллиптической кривой, также известную как аддитивная группа точки на кривой. Для того, чтобы вывести общие формулы для сложения точек зададим кривую, уравнение которой выглядит так: $y^2=x^3+ax+b$» [24].

Для того, чтобы найти сумму двух разных точек, которые лежат на эллиптической кривой, нужно определиться с несколькими моментами. Представим, что у нас есть две точки P и Q с координатами x_1, y_1 и x_2, y_2 соответственно. Для начала нужно вычислить наклон, он необходим для дальнейших вычислений и являет собой угол наклона прямой. Является либо касательной к точке кривой, либо соединяет две разные точки. Наклон представлен в формуле (7).

$$\lambda = \frac{y2-y1}{x2-x1}. \quad (7)$$

«Затем вычисляются координаты x_3 и y_3 для определения новой точки R на эллиптической кривой» [11]. Вычисляем координаты x_3 по формуле с

помощью квадрата наклона и последующего вычитания первой координаты x_1 и второй координаты x_2. Процесс представлен в формуле (8).

$$x_3=\lambda_2-x_1-x_2. \quad (8)$$

Координаты y_3 вычисляются по-другому. Умножаем наклон на разность координат x_1 и x_3 вычитаем координату y_1. Процесс представлен в формуле (9).

$$y_3=\lambda(x_1-x_3)-y_1. \quad (9)$$

В конечном итоге получим новую точку R с координатами x_3 и y_3.

Рассмотрим следующую операцию на эллиптических кривых. «Предположим, что у нас точка P равна точке Q, тогда мы можем удвоить данную точку, получив в итоге новую точку R на кривой. Здесь мы будем использовать координаты точки P, так как координаты точки Q аналогичные. Для начала нужно определить наклон» [8]. Теперь он будет иметь другую формулу, в которой будет использован (a) коэффициент эллиптической кривой. Коэффициент (a) влияет на наклон касательной к кривой в любой точке P, которую мы используем для данных вычислений, а также влияет на положение кривой на плоскости. Наклон для последующего удвоения представлен в формуле (10).

$$\lambda = \frac{3x1^2}{2y1}. \quad (10)$$

Далее нужно вычислить координаты x_3 точки R. Для этого надо вычесть удвоенную координату x_1 из квадрата наклона. Применяем следующую формулу (11)

$$x_3=\lambda^2-2x_1. \quad (11)$$

Координата y_3 вычисляется аналогично, «как и в методе сложения точек. Умножаем наклон на разность координат x_1 и x_3 вычитаем координату y_1» [14].

В итоге получаем все координаты для новой точки R на нашей эллиптической кривой.

Также можно рассмотреть особый случай, когда точка P равна отрицательной точке P. При сложении таких точек мы получаем точку на бесконечности O, аналогичную нулю в обычной арифметике. Представим формальное доказательство. «Пусть P будет с координатами x и y, а –P с координатами x и -y, тогда сложение этих точек можно записать как P+(-P)=O. Так как P и –P имеют одинаковую абсциссу x, прямая, проходящая через эти точки вертикальна. В контексте аддитивной группы точек на эллиптической кривой, пересечение такой прямой с кривой и определение суммы точек ведёт к нейтральному элементу O» [18].

В заключение можно сказать, что сложение точки с ее обратной всегда приводит к образованию нейтрального элемента аддитивной группы, точке на бесконечности O. Это ключевой аспект «аддитивной группы точек на эллиптической кривой и обеспечивает корректность криптографических операций, таких как ECDH и ECDSA» [6].

1.1.4 Умножение точки на скаляр

«Скалярное умножение представляет собой простую операцию в виде умножения точки на целое число» [6].

«Для выполнения скалярного умножения на эллиптической кривой используется метод двоичного разложения скаляра. Этот метод позволяет эффективно вычислить произведение точки на эллиптической кривой на заданное целое число.

Кратко, алгоритм скалярного умножения на эллиптической кривой с использованием метода двоичного разложения выглядит примерно следующим образом» [11]:

– представим скаляр k в двоичном виде. Например, если k = 19, то в двоичной системе: k = 10011;
– начнём с пустой точки (нейтральный элемент) и для каждого бита b_i в двоичном представлении k выполним следующие шаги;
– удвоим текущую точку (выполнение операции сложения с самим собой);
– Если b_i= 1, добавим исходную точку к текущей точке.

Этот процесс позволяет эффективно вычислить скалярное умножение точки на кривой на заданное целое число.

«Метод "удвой и добавь" (или "удвоение и сложение") задействуется для вычисления скалярного умножения на эллиптических кривых, который используется для эффективного вычисления кратной точки на кривой по заданному целому числу» [23]. Метод представлен на рисунке 3.

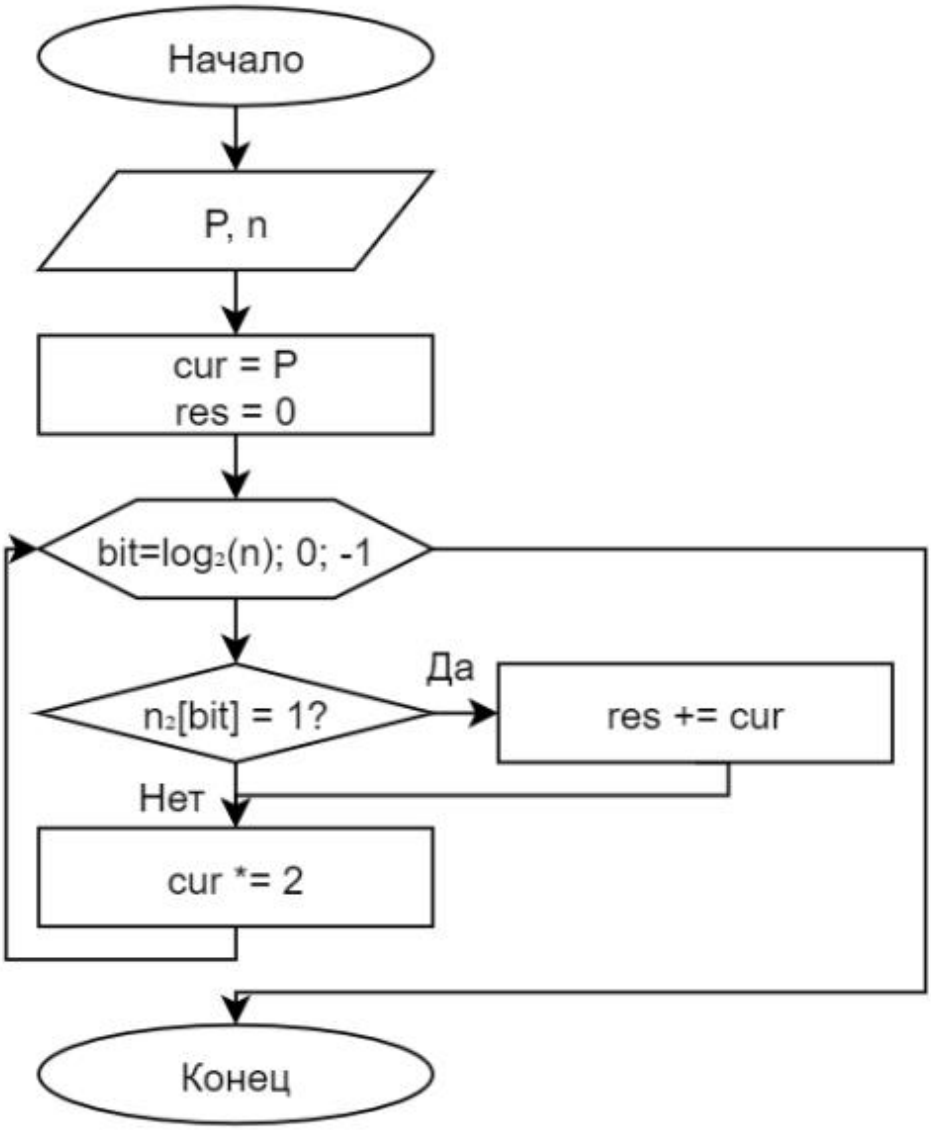

Рисунок 3 - Метод "удвой и добавь"

«Этот метод применяется при работе с эллиптическими кривыми в криптографических приложениях, таких как цифровая подпись или шифрование» [10]. Он основан на математических свойствах эллиптических кривых и обеспечивает эффективный способ выполнения умножения точки кривой на целое число.

Алгоритм "удвой и добавь" работает следующим образом:

– начнём с пустой точки (нейтральным элементом) и исходной точки на эллиптической кривой.
– представим скаляр (целое число) в двоичной форме.
– используя двоичное представление скаляра, последовательно выполним следующие действия:
– удвоение: удваиваем текущую точку на кривой, что эквивалентно выполнению операции сложения точки самой с собой.
– добавление: если очередной бит скаляра равен 1, то добавляем исходную точку к текущей точке.

Эти шаги повторяются для всех битов двоичного представления скаляра. После завершения этого процесса, конечная точка будет результатом скалярного умножения исходной точки на заданное целое число.

«Метод "удвой и добавь" обеспечивает эффективный способ вычисления скалярного умножения на эллиптических кривых и широко применяется в реализации криптографических примитивов, использующих эллиптические кривые» [3].

1.1.5 Задача логарифмирования на эллиптической кривой

«Задача логарифмирования в криптографии относится к вычислению дискретного логарифма в группе. Для конкретики рассмотрим группу мультипликативной подгруппы конечного поля или подгруппы эллиптической кривой» [24]. Задача логарифмирования здесь заключается в нахождении такого числа x, что для заданных значений g и y выполняется равенство $g^x = y$.

Задача логарифмирования имеет важное значение в криптографии, поскольку многие современные криптографические системы базируются на сложности решения этой задачи. Например, в криптосистеме Диффи-Хеллмана или метода шифрования Эль-Гамаля, сложность нахождения дискретного логарифма используется для обеспечения безопасности передачи данных.

В криптографии существует два основных типа задачи логарифмирования: вычисление дискретного логарифма (DLP) и «задача логарифмирования на эллиптических кривых (ECDLP). Оба эти типа являются сложными задачами вычисления, особенно при больших значениях модуля или порядка эллиптической кривой» [17].

«Решение задачи логарифмирования в общем случае требует значительных вычислительных ресурсов и может быть сложной в реализации. Исследование алгоритмов для решения задачи логарифмирования и поиск криптографически стойких методов защиты от этой атаки является активной областью исследований в сфере криптографии» [5].

1.2 Конечные поля при эллиптические кривых

1.2.1 Виды полей над кривыми

Для того чтобы лучше разобраться в эллиптических кривых, нужно ввести понятие поля. Поле – это множество F с несколькими операциями, такими как сложение и умножение или же аддитивная и мультипликативная операции соответственно, при условии, что $1 \neq 0$, то есть поле F образует коммуникативную группу по сложению, и коммуникативную группу по умножению, которая образовывается из всех ненулевых элементов.

«К характеристикам поля можно отнести следующее: если наименьшее положительное число n такое, что сумма n копий единицы равна нулю, то $n \cdot 1 = 0$. Если такого числа не существует, тогда характеристика равна нулю» [21].

Также существует еще одно понятие, можно сказать, что «существует расширение поля, которое означает поле, содержащее данное поле в качестве подполя» [10].

Поле Галуа – это множество, состоящее из конечного числа элементов, на котором определены операции сложения, вычитания, умножения и деления, удовлетворяющие обычным аксиомам поля. Так как поля конечны, их легко использовать в компьютерных системах, потому что дан фиксированный набор значений. Операции сложения точек выполняются именно в таком поле.

Основные свойства конечных полей:

- конечное число элементов: «количество элементов в поле конечно и равно p^n, где p – простое число, называемое характеристикой поля, а n представляет собой положительное целое число» [7];
- замкнутость: результаты операций сложения, вычитания, умножения и деления (кроме деления на ноль) всегда являются элементами поля;
- ассоциативность, коммутативность и дистрибутивность: «Операции в поле обладают свойствами ассоциативности, коммутативности и дистрибутивности» [3];
- существование нейтральных элементов: в поле существуют нейтральные элементы для сложения 0 и умножения 1;
- существование обратных элементов: для каждого элемента a поля, кроме 0, существует обратный элемент a^{-1} такой, что при умножении элемента на обратный результат будет равен единице.

«Очевидно, что обязательные свойства поля включают в себя все обязательные свойства абелевой группы. Следовательно, любое поле также является абелевой группой и сохраняет все свойства группы» [10].

Число p должно обязательно являться простым, иначе не будут выполняться все свойства поля.

1.2.2 Поле F_p и эллиптические кривые над ним

В криптографических системах кубическая кривая обычно рассматривается надо конечным полем Галуа GF(p) или $GF(p^n)$, где p – простое число. Здесь все операции выполняются по модулю p.

Эллиптические кривые обеспечивают высокую степень безопасности при относительно небольших длинах ключей, что делает их привлекательным выбором для многих криптографических систем, над конечными полями они являются важной составляющей современной криптографии и широко применяются в различных криптографических алгоритмах» [8].

Ограниченная конечным полем F_p эллиптическая кривая задана в формуле (12).

$$\{(x, y) \in Fp\ 2 \vee y\ 2 \equiv x\ 3 + ax + b(modp), 4a\ 3 + 27b\ 2 \not\equiv \not\equiv 0(modp)\} \cup \{0\}, \quad (12)$$

где 0 – бесконечно удаленная точка,

a, b – целые числа, принадлежащие F_p.

Вид над конечным полем, с заданными параметрами, представлен на рисунке 4.

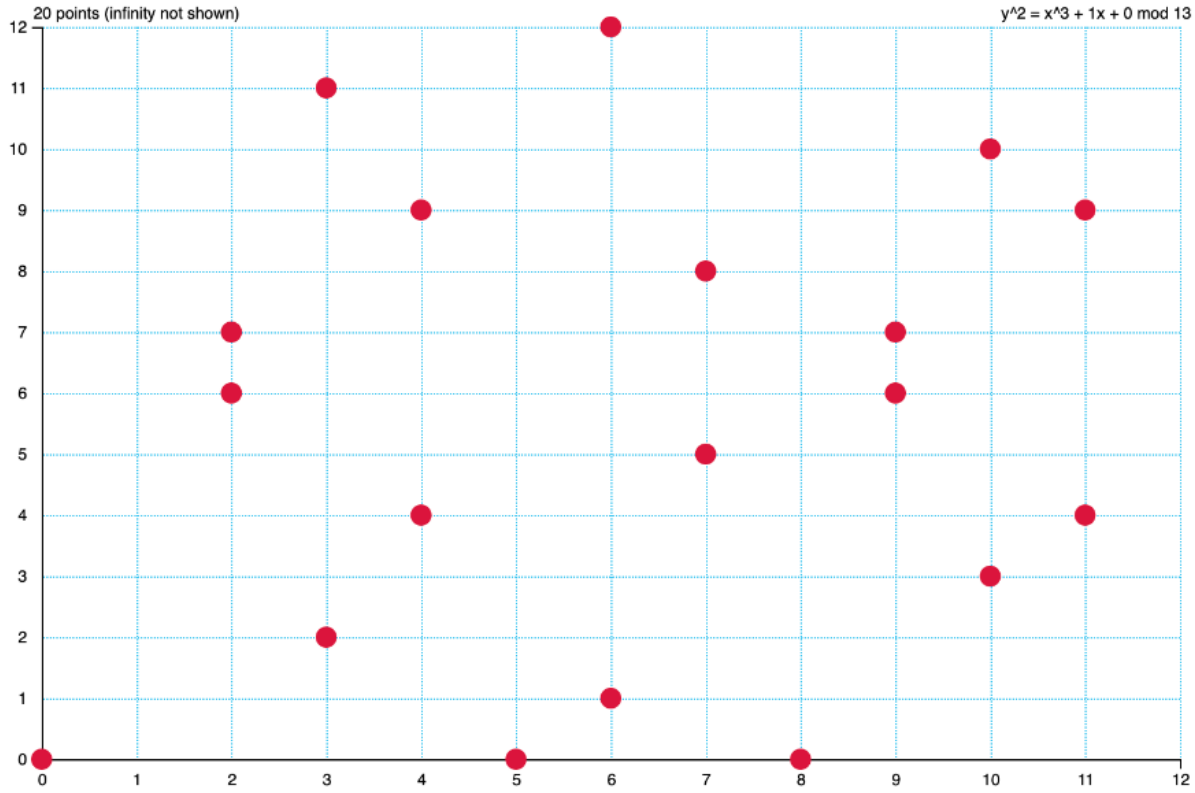

Рисунок 4 – Конечное поле и кривая на нем

«Когда эллиптическая кривая рассматривается над конечным полем, ее гладкая форма трансформируется в множество изолированных точек, размещенных на регулярной решетке. Несмотря на видимую дискретность, эта кривая сохраняет все свои основные свойства» [19]. Графическое представление также отражает симметрию, присущую данным кривым, которая проявляется относительно прямой y = p/2, что является характерным признаком для кривых в контексте модулярной арифметики.

В заключение можно сказать, что эллиптические кривые над полем Fp являются мощным инструментом в современной криптографии, обеспечивая высокий уровень безопасности при меньших размерах ключей, по сравнению с традиционными алгоритмами. Понимание их математических основ и операций позволяет эффективно использовать их в различных криптографических приложениях.

1.2.3 Операция сложения точек над конечным полем

Сложение точек на эллиптической кривой над конечным полем является одним из основных и важных методов в алгебре эллиптических кривых. При сложении можно найти сумму двух точек на кривой, результатом этого сложения станет новая точка. Этот метод является основой для выполнения других операций, таких как удвоение точки или умножение точки на целое число.

Для того, чтобы найти сумму двух точек на эллиптической кривой, сначала необходимо проверить, находятся ли эти точки на одной и той же кривой. «Если нет, то результатом сложения будет элемент, называемый бесконечно удаленной точкой, затем нужно определить линию, которая проходит через наши точки P и Q, которые по определению не могут быть равны. Точка R здесь не присутствует, а это значит, что прямая касательна к кривой, так как проходит через две точки. Если наша линия пересекает кривую в третьей точке, а так происходит всегда на эллиптической кривой, то эта точка является суммой двух данных точек» [20]. Найденная сумма будет третьей точкой на эллиптической кривой, обычно обозначаемой как R.

«Несмотря на то, что алгоритм сложения точек на эллиптической кривой над конечным полем может быть сложным, оно является основой для многих протоколов криптографии, таких как алгоритмы обмена ключами Диффи-Хеллмана (ECDH) и криптографических подписей на эллиптических кривых (ECDSA)» [8].

Сложение точек на эллиптической кривой над конечным полем обладает рядом интересных свойств. Например, сложение точек обладает ассоциативностью, то есть (P + Q) + R = P + (Q + R), что позволяет нам складывать точки с использованием нескольких операций подряд и получать один и тот же результат. Также существует нейтральный элемент относительно сложения, который называется нулевой (или бесконечно

удаленной) точкой, и он действует как элемент "поглощения", то есть P + O = P.

«Есть особые случаи при выполнении сложения точек на эллиптической кривой над конечным полем. Когда мы находим произведения двух точек P и Q мы должны быть аккуратными с коэффициентами уравнения эллиптической кривой и особыми случаями, такими как случай, когда точки совпадают или когда линия, проходящая через них будет вертикальной» [12]. В этих ситуациях применяются специальные формулы, которые позволяют правильно выполнить сложение. Особые случаи, это сумма точки и ее обратной и сумма точки и тоски на бесконечности, а также удвоение точки.

Также стоит отметить, что сложение точек на эллиптической кривой является вычислительно интенсивной операцией, особенно при работе с большими числами или большими полями, поэтому «были разработаны различные алгоритмы и оптимизации для ускорения вычислений, такие как метод двоичного разложения или умножение Карацубы для больших чисел» [17].

Данная операция над конечным полем играет важную роль в криптографии, так как обеспечивает безопасность и эффективность различных криптографических протоколов. Это связано с трудностью обратного процесса вычисления, то есть нахождения исходных точек при известной сумме. Эта сложность делает эллиптические кривые привлекательными для использования в криптографии, особенно при реализации схемы обмена ключами Диффи-Хеллмана, точнее более усовершенствованного алгоритма на основе этого, или криптографических подписей на эллиптических кривых.

1.2.4 Порядок группы на эллиптической кривой

«Общее количество точек, принадлежащее кривой, включающее точку на бесконечности O, называется порядком подгруппы» [1].

Для того чтобы вычислить порядок эллиптической кривой над конечным полем Fp, где p – простое число, нужно использовать алгоритм, известный как алгоритм Шуфа. Он позволяет определить точное количество точек на кривой, включая точку на бесконечности.

Алгоритм Шуфа включает в себя следующие шаги:

– инициализируем счетчик точек N на 1, который нужен для учета точки на бесконечности;
– добавляем цикл по всем возможным значениям x из конечного поля Fp;
– для всех x вычисляем соответствующее значение y^2 по модулю p при помощи уравнения эллиптической кривой $y^2 = x^3 + ax + b(\mathrm{mod}\ p)$; Использование mod p нужно для определения кривой над конечным полем;
– если наше полученное значение y^2 является квадратом Fp, то существуют две точки на кривой с данной координатой x и счетчик N увеличивается на 2;
– после завершения цикла по x, N будет представлять собой порядок эллиптической кривой.

Стоит упомянуть, что порядок группы должен быть большим, чтобы обеспечить надежную безопасность в криптографических системах. «Для гарантии того, что проблема дискретного логарифмирования является сложной задачей, в криптографии на эллиптических кривых используют порядки» [9].

«Кроме того, порядок группы кривой связан с ее порядком подгрупп через теорему Лагранжа. "Она утверждает, что порядок любой подгруппы H группы G является делителем порядка группы G.В нашем случае это означает, что порядок циклической подгруппы, образованной точкой G, должен быть делителем общего порядка кривой» [4].

1.2.5 Подгруппы на эллиптической кривой

Циклические подгруппы являются подмножествами элементов, которые образуют группу на эллиптической кривой. Любой элемент подгруппы может быть получен путем повторного применения групповой операции к одному элементу, который называется генератором. Циклическая подгруппа состоит из точек на кривой, полученных путем сложения генератора с самим собой. Этот процесс является основным действием в эллиптической криптографии.

Основные характеристики циклических подгрупп:

– генератор присутствует в каждой циклической подгруппе. Он является важным элементом, который путем умножения скаляра на себя получает все остальные элементы подгруппы;
– порядок подгруппы представляет собой количество всех элементов. Является наименьшим n – положительным, таким, что n – кратное умножение генератора на себя даст нейтральный элемент группы. Например, $nG = O$, где O – точка на бесконечности;
– согласно теореме Лагранжа, порядок любой подгруппы является делителем порядка группы;
– проблема дискретного логарифма возникает в циклической подгруппе, которая заключается в нахождении целого числа k, если даны генератор g и g^k в подгруппе;
– кофактор группы является отношением порядка группы к порядку ее подгруппы. Он служит для определения, насколько группа является большой по отношению к ее подгруппе. Если кофактор слишком малый, тогда это указывает на потенциальные уязвимости, связанные с малыми подгруппами.

Предположим, что у нас есть точка P на эллиптической кривой. Тогда с помощью теоремы Лагранжа можно составить алгоритм для нахождения порядка подгруппы:

– «сначала вычислим порядок эллиптической кривой, используя алгоритм Шуфа» [21];
– начнем с точки P. Инициализируем счетчик n =1;
– умножим точку P на n и проверим, равно ли полученное значение точке на бесконечности O;
– Если nP = O, тогда n будет являться порядком подгруппы с базисом в точке P. Если нет, увеличим n на единицу и повторим шаг с умножением на точку P. Так мы найдем порядок;
– «после того как определили порядок подгруппы, проверяем с помощью теоремы Лагранжа, что n действительно делит порядок кривой.

Этот алгоритм основан на поиске наименьшего положительного целого числа n, такого что $nP = O$, что является определением порядка подгруппы с базисной точкой P» [25].

1.2.6 Задача дискретного логарифмирования на эллиптической кривой

Задача дискретного логарифмирования на эллиптических кривых заключается в поиске такого целого числа k, что умножение точки P на это число k даёт другую точку Q на эллиптической кривой, где kP означает точку P сложенную с собой k раз.

«Формально, задачу можно сформулировать следующим образом: пусть даны точки P и Q на эллиптической кривой E, и известно, что точка Q равна k_P для некоторого целого числа k» [11]. Требуется найти значение этого числа k.

Решение таких примеров дискретного логарифмирования на эллиптических кривых является сложной задачей и используется в криптографии для построения криптосистем, таких как алгоритм обмена ключами (ECDH) и цифровая подпись с открытым ключом (ECDSA). Для поиска дискретного логарифма на эллиптической кривой существуют

различные алгоритмы, такие как алгоритм Полларда, метод индексного и алгоритмы на основе рациональных точек.

«Эллиптические кривые и связанные с ними задачи, такие как ECDLP, играют ключевую роль в современной криптографии, несмотря на их сложность и глубокие математические основы, понимание принципов работы эллиптических кривых и задачи дискретного логарифмирования на них является ключом к пониманию многих современных криптографических систем» [7].

Выводы по итогам первой главы: несингулярность кривой один из главных признаков для алгоритмов, группа точек является необходимым условием для работы методов, для создания эллиптической кривой необходимо ввести понятие конечного поля, операция сложения точек является самым важным методом для криптографических алгоритмов.

Глава 2 Алгоритмы шифрования на эллиптических кривых

2.1 Алгоритм обмена ключами

Шифрование данных используется для преобразования информации в зашифрованный вид, который не может быть расшифрован без специального ключа или пароля. Оно используется для конфиденциальности информации.

Так как проблема безопасного использования ключа является важнейшим вопросом в криптографии, мы рассмотрим алгоритм, который обеспечивает безопасную передачу ключей.

Алгоритм обмена ключами ECDH нужен для обеспечения безопасности при обмене секретными ключами. Так две стороны (два человека) могут сделать обмен по открытому каналу без рисков. Алгоритм обеспечивает конфиденциальность при обмене.

Работу алгоритма я разбил на 4 шага для наглядности:

Представим этот алгоритм графически:

- для начала нужно выбрать эллиптическую кривую, например SECP256k1, а точнее ее параметры, такие как генератор, конечное поле и порядок кривой;
- далее идет генерация ключей. «У каждой стороны есть по два ключа, приватный ключ, который является случайным числом, выбранным из большого диапазона значений и публичный ключ, получаемый путем умножения приватного ключа на заданный генератор эллиптической кривой» [4];
- затем происходит обмен публичными ключами по открытому каналу связи;
- после этого вычисляем общий секретный ключ для шифрования, дешифрования и аутентификации. Предположим, что у человека A есть ключи a и A, а у человека B есть ключи b и B, обозначающие

приватные и публичные ключи соответственно. Тогда вычисление секретного ключа происходит по формуле $S=B\cdot a=(b\cdot G)\cdot a=b\cdot(a\cdot G)=A\cdot b$, где G является генератором на эллиптической кривой, то есть выбранной точкой для обоих участников обмена.

Алгоритм обмена ключами ECDH представлен на рисунке 5.

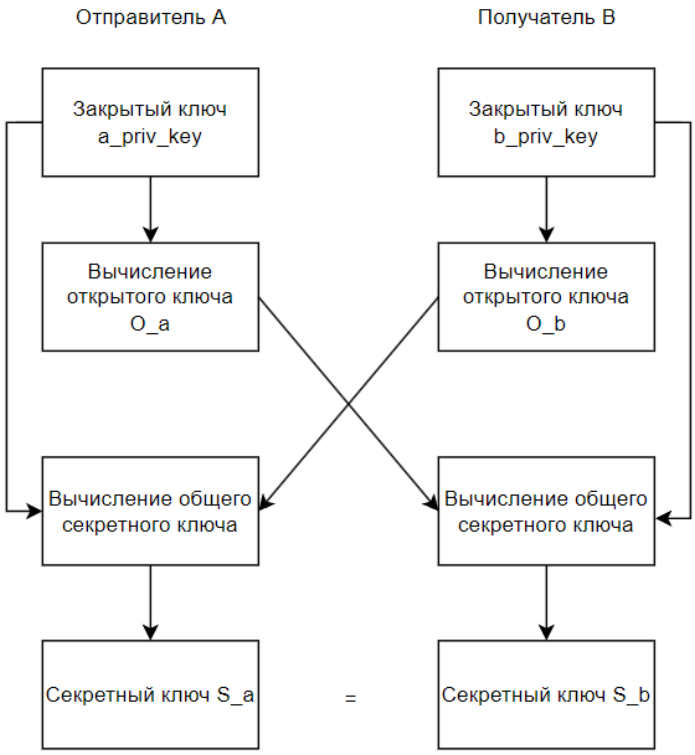

Рисунок 5 – Алгоритм ECDH

Алгоритм ECDH отличается от других высокой безопасностью, из-за использования задачи дискретного логарифмирования. Также этот алгоритм имеет меньший ключ по сравнению с другими системами шифрования.

Благодаря обмену ключами можно создать безопасный секретный ключ для дальнейшего использования в криптографических системах.

2.2 Алгоритм цифровой подписи

«Алгоритм ECDSA (Elliptic Curve Digital Signature Algorithm) – это реализация схемы цифровой подписи, основанная на использовании эллиптических кривых и модульной арифметики» [6].

«Эллиптическая кривая в ECDSA – это линия на плоскости, задаваемая уравнением $y^2=x^3+a\cdot x+b$, где *a* и *b* – такие числа, что $4\cdot a^3+27\cdot b^2\neq 0$» [7].

Первым делом мы выбираем кривую и ее параметры, наилучшим вариантом будет кривая SECP256k1, которую я приводил в пример при разборе алгоритма обмена ключами, затем мы генерируем публичный и приватный ключи, для начала генерируем закрытый ключ, который является случайным набором символов, далее получаем открытый ключ путем умножения приватного ключа на точку эллиптической кривой, которую выбрали оба участника обмена. Теперь рассмотрим разные варианты, для этого нам понадобится точка k, которая является целым положительным числом, и точка G. Здесь может быть несколько альтернатив, например, если при умножении G на k мы получим ту же точку G, тогда положение точки на кривой не изменится, если точка не совпала с G, но всё равно лежит на кривой, значит найденная точка скользит по кривой. Также может быть, что наш элемент будет равен точке на бесконечности. В конце реализуем алгоритм создания подписи, где на вход мы принимаем приватный ключ PrivateKEY и число m – хэш подписываемого сообщения [6]. Параметры подписи r и s вычисляются как раз с помощью того, что мы приняли на вход, r получаем с помощью k, а s находим с помощью PrivateKEY, k и m.

Проведем верификацию полученной подписи с помощью нескольких шагов:

– проверим по формуле правильность вычислений:

  $1 \leq r \leq n-1$ и $1 \leq s \leq n-1$. Если всё верно, то можно переходить к следующему шагу;

– затем вычислим параметр w, уравнение которого выглядит так:
  $w = s^{-1} \bmod n$. Этот параметр понадобится нам для нахождения координат новой точки;
– далее найдем параметры u_1 и u_2, они нужны как координаты третьей точки на нашей кривой:
  $u1 = hash(message) \cdot w \bmod n$;
  $u2 = r \cdot w \bmod n$;
– в конце получим новую точку Q' равную $u_1 \cdot G + u_2 \cdot public\_key$;
– теперь наша подпись будет действительна, если x-координата точки Q' (r) равна r.

Графически алгоритм представлен на рисунке 6:

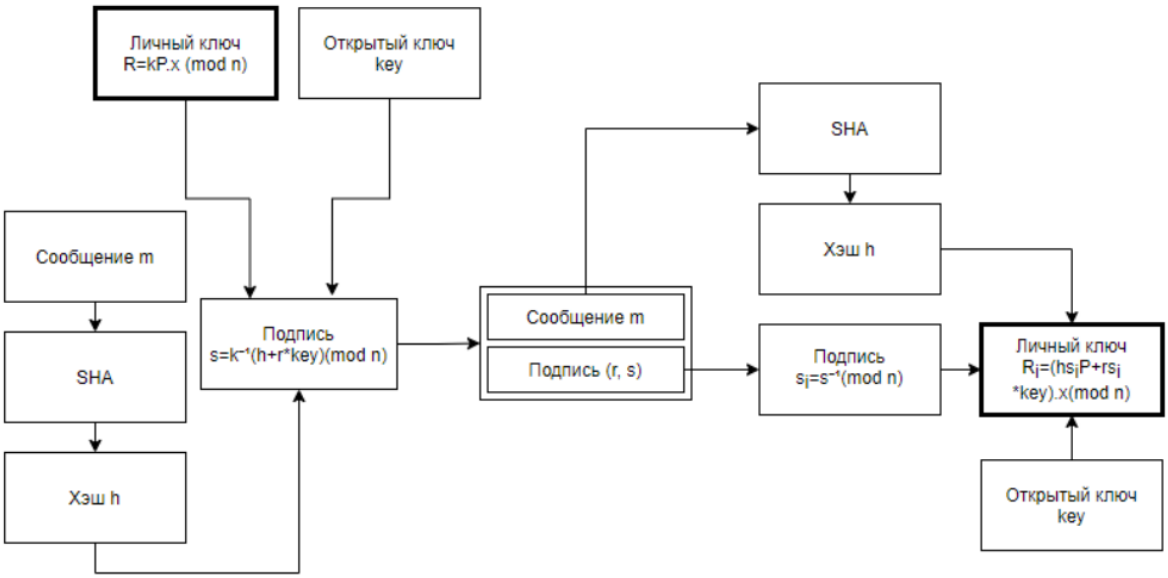

Рисунок 6 – Алгоритм ECDSA

Из всего вышеперечисленного можно сделать вывод, что «алгоритм цифровой подписи ECDSA является важным в криптографии, так как обеспечивает безопасность и целостность передаваемой информации» [11].

2.3 Смешанные алгоритмы

ECIES (Elliptic Curve Integrated Encryption Scheme) – это схема шифрования, которая объединяет асимметричное шифрование на основе эллиптических кривых с симметричными алгоритмами шифрования и хэш-функциями для создания защищённого и эффективного метода шифрования. «В ECIES отправитель генерирует случайный симметричный ключ и использует его для шифрования сообщения с помощью симметричного алгоритма (например, AES). Затем этот симметричный ключ шифруется с использованием открытого ключа получателя, основанного на эллиптических кривых. Хэш-функции используются для генерации ключей и проверки целостности данных» [9]. Получатель, имея соответствующий закрытый ключ, может расшифровать симметричный ключ, а затем использовать его для расшифровки самого сообщения, обеспечивая таким образом конфиденциальность и целостность обмена данными.

Алгоритмы ECIES представлены в виде нескольких шагов:

– для начала приступим к шифрованию с открытым ключом, «для этого используем алгоритм шифрования KEM, задаем параметры кривой и получаем симметричный ключ, алгоритм позволит нам зашифровать данные с использованием открытого ключа» [13];
– далее используем алгоритм хэширования, чтобы обеспечить безопасность и целостность данных, для хэширования наиболее подходящим вариантом будет хэш-функция SHA-256, обладающая криптографической стойкостью;
– затем идет генерация ключей с помощью эллиптической кривой. Нужно сгенерировать приватный ключ и с его помощью публичный ключ;
– также нужно получить производные ключи для последующей аутентификации или управления сеансовыми ключами.

Шифрование ECIES происходит по следующим этапам:

– сначала проводим инициализацию, то есть задаем параметры нашей эллиптической кривой, такие как коэффициенты и порядок. Также зададим базовую точку G и порядок подгруппы кривой;
– сгенерируем закрытый и открытый ключи d_A и $Q_A=d_A \cdot G$;
– далее, после получения публичного ключа Q_B, выбираем случайное число k и высчитываем точку C равную произведению k на G.
– затем, получаем секретный ключ r равный произведению открытого ключа на закрытый ключ;
– в конце происходит шифрование данных, для которого мы используем найденный секретный ключ, с помощью которого получаем ключ шифрования, и шифруем данные с помощью симметричного алгоритма AES. Здесь нужно добавить вектор инициализации для повышения безопасности.

Алгоритм представлен на рисунке 7:

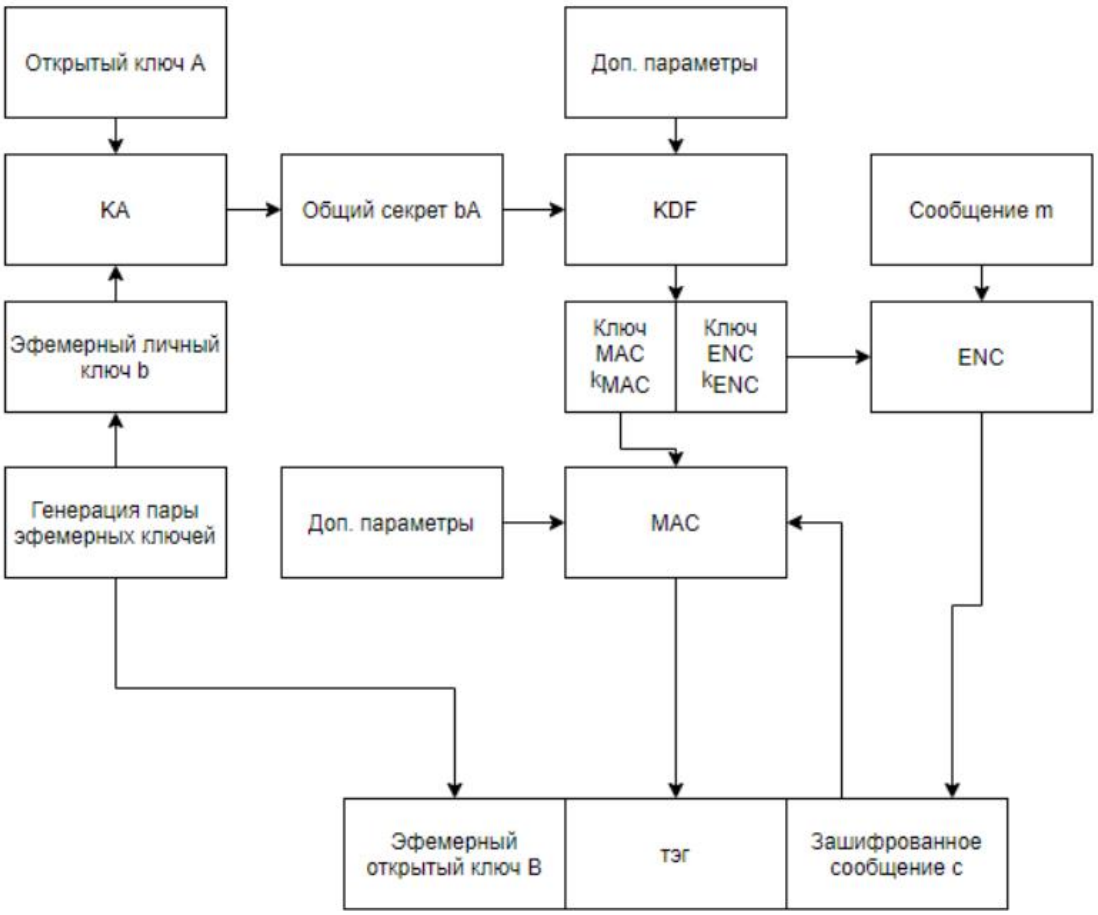

Рисунок 7 – Алгоритм ECIES

Таким образом, ECIES обеспечивает надежное шифрование данных с использованием смешанных алгоритмов.

Выводы по итогам второй главы: алгоритм обмена ключами имеет меньший ключ по сравнению с другими алгоритмами шифрования, алгоритм цифровой подписи обеспечивает безопасность и целостность передаваемой информации, Смешанные алгоритмы отличаются надежным шифрованием данных.

Глава 3 Программная реализация алгоритмов шифрования

3.1 Выбор средств

«Для реализации криптографического алгоритма на эллиптических кривых был выбран язык python» [2].

Для выполнения работы понадобится библиотека ecdsa, которая предоставляет функциональность для работы с эллиптическими кривыми и цифровыми подписями на их основе.

Основные особенности библиотеки:

- поддерживает несколько стандартных эллиптических кривых, включая популярные кривые, такие как SECP256k1, SECP256r1, и другие;
- «ecdsa выполняет ключевой обмена Диффи-Хеллмана на основе эллиптических кривых» [20], что позволяет двум сторонам безопасно генерировать общий секретный ключ;
- ключи и подписи, созданные с помощью ecdsa, могут быть использованы с другими библиотеками и системами, поддерживающими стандарты эллиптических кривых. Это облегчает интеграцию с другими криптографическими системами;
- библиотека реализует проверенные криптографические алгоритмы, обеспечивая высокий уровень безопасности.

3.2 Структура проекта

В работе будут задействованы классы и функции. «Класс в программировании – это модель для создания объектов определённого типа» [2]. Он описывает структуру объектов (набор полей и их начальное состояние) и определяет алгоритмы (функции или методы) для работы с

этими объектами, класс служит средством для введения абстрактных типов данных в программный проект. «Функции – это блоки кода, выполняющие определенные операции. Если требуется, функция может определять входные параметры, позволяющие вызывающим объектам передавать ей аргументы» [4]. При необходимости функция также может возвращать значение как выходное.

3.3 Реализация методов и операций на эллиптических кривых

Эллиптические кривые я разрабатывал через иерархию классов и функций внутри них. «Были разработаны 4 различных класса для подробного описания работы методов на эллиптических кривых и отдельный класс для реализации алгоритма ECDSA» [18].

– TFF – для конечного поля и методов внутри него;
– ECpoint – точка на эллиптической кривой;
– EC – эллиптическая кривая;
– ECgroup – группа над полем и методы внутри этой группы.

В классе TFF, который является конечным полем, реализованы методы для наглядности работы эллиптических кривых. Эти методы так или иначе используются в алгоритмах шифрования. Теперь можно объяснить зачем нужен каждый из этих методов.

Метод inv нужен для получения обратной точки на эллиптической кривой.

Метод addINV нужен для суммирования с найденным обратным элементом.

Метод neg для нахождения отрицания.

Метод add для сложения точек эллиптической кривой.

Метод scalar_mult для скалярного умножения точки.

Все методы вместе с полем представлены на рисунке 8.

```python
class TFF:
    def __init__(self, p, m):
        self.p = p
        self.m = m
        self.Fp = GF(p)
        self.x = symbols('x')
        self.irred = poly(self.x**m + self.x + 1, modulus=p)

    def add(self, a, b):
        return (a + b) % self.p

    def neg(self, a):
        return (-a) % self.p

    def inv(self, a):
        return pow(a, -1, self.p)

    def addInv(self, a, b):
        return (a - b) % self.p

    def div(self, a, b):
        return (a * self.inv(b)) % self.p

    def scalarMult(self, a, k):
        return (a * k) % self.p
```

Рисунок 8 – Конечное поле и методы внутри него

Далее рассмотрим класс ECpoint. С помощью него можем определить, лежат ли действительно точки на кривой, у этого класса определены лишь элементы x и y отвечающие координатам на эллиптической кривой. С помощью него можно будет создать точки для дальнейшего применения.

Программная реализация класса ECpoint представлена на рисунке 9.

```python
class ECpoint:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return f"ECpoint({self.x}, {self.y})"
```

Рисунок 9 – Класс Ecpoint

Этот класс служит для нахождения точки.

Теперь рассмотрим класс EC. Он нужен для определения эллиптической кривой, содержит три элемента отвечающие трем параметрам кривой. «Здесь a и b – коэффициенты эллиптической кривой, а p – поле, над которым лежит кривая» [16]. Этот класс служит для создания точек на эллиптической кривой.

Реализация представлена на рисунке 10.

```
class EC:
    def __init__(self, a, b, p):
        self.a = a
        self.b = b
        self.p = p
        self.curve = CurveFp(p, a, b)

    def is_on_curve(self, pt):
        return self.curve.contains_point(pt.x, pt.y)
```

Рисунок 10 – Класс EC.

Этот класс представляет собой эллиптическую кривую с проверкой принадлежности точки.

Рассмотрим последний класс под называнием ECgroup. Он содержит все необходимые методы и операции на кривых. Таблица класса представлена на рисунке. «Именно этот класс понадобится для дальнейшего вывода результатов работы на эллиптических кривых, таких как сложение, удвоение точки, скалярное умножение, обратная точка и проверка на обратность к данной точке» [19].

Класс состоит из методов. Рассмотрим каждый из них подробнее.

Первый метод add – сложение точек на эллиптической кривой

Второй метод scalar_mult – умножение точки на скаляр.

Третий метод inv – нахождение обратной точки.

Четвертый метод isInv – проверка является ли точка обратной к другой точке на данной кривой.

Пятый метод Double – удвоение точки.

Шестой и последний Compute – для сложения двух скалярных произведений.

Этот класс позволяет использовать методы на эллиптических кривых.

Программная реализация класса группы точек кривой ECgroup представлен на рисунке 11.

```
class ECgroup:
    def __init__(self, curve, generator, order):
        self.curve = curve
        self.generator = generator
        self.order = order

    def add(self, pt1, pt2):
        return pt1 + pt2

    def scalar_mult(self, k, pt):
        return k * pt

    def inv(self, pt):

        return Point(self.curve, pt.x(), -pt.y() % self.curve.p())

    def isInv(self, pt1, pt2):

        return self.inv(pt1) == pt2

    def Double(self, pt):

        return pt + pt

    def compute(self, k1, pt1, k2, pt2):

        return (k1 * pt1) + (k2 * pt2)
```

Рисунок 11 – Класс Ecgroup

Данный класс отвечает за методы и операции на кривой.

Здесь я описал классы, которые реализуют все важные методы на эллиптических кривых.

3.4 Реализация алгоритма обмена ключами

Алгоритм состоит из одной функции. На вход подается O_a и O_b взятые из класса ECpt, а также ECgroup для реализации скалярного умножения scalar_mult.

Приватные ключи генерируются с помощью функции SigningKey библиотеки ecdsa. В коде использована предопределённая эллиптическая кривая SECP256k1, предоставляемая библиотекой ecdsa. O_a и O_b используются для получения значений открытых ключей. Такой ключ получается при умножении приватного ключа a или b умножением на одну и ту же точку на эллиптической кривой “Point”. Потом путем скалярного умножения приватный ключ a умножаем на открытый ключ b и приватный ключ b умножаем на открытый ключ a, получаем общий секретный ключ.

Функция обмена ключами представлена на рисунке 12.

```
def ECDH(ecgroup, O_a, O_b):

    a_priv_key = SigningKey.generate(curve=SECP256k1)
    b_priv_key = SigningKey.generate(curve=SECP256k1)

    O_a = a_priv_key.verifying_key.pubkey.point
    O_b = b_priv_key.verifying_key.pubkey.point

    S_a = ecgroup.scalar_mult(a_priv_key.privkey.secret_multiplier, O_b)
    S_b = ecgroup.scalar_mult(b_priv_key.privkey.secret_multiplier, O_a)

    assert S_a == S_b

    return S_a
```

Рисунок 12 – Реализация функции обмена ключами

Данная функция позволит получить общий секретный ключ для обоих участников обмена.

SECP256k1 – это стандартная кривая, используемая в криптографии на основе эллиптических кривых (ECC). Она широко известна благодаря своему применению в биткоине и других криптовалютах. Кривая SECP256k1 обеспечивает высокий уровень безопасности благодаря 256-битному ключу, что делает атаки методом полного перебора практически невозможными с текущими вычислительными мощностями.

3.5 Реализация цифровой подписи

Цифровая подпись состоит из класса ECDSA, в котором производятся основные вычисления и функция demo_ecdsa, которая демонстрирует работу алгоритма.

Для последующей работы мы должны определить кривую. Как и в ECDH я выбрал SECP256k1, благодаря ее параметрам безопасности.

Класс ECDSA состоит из нескольких функций:

- init для определения кривой
- generate_keys для генерации приватного и публичного ключа. Приватный ключ генерируется с помощью SigningKey библиотеки ecdsa. Публичный ключ получаем с помощью приватного ключа с использованием verifying_key;
- sign для вычисления хэша сообщения и подписи. Здесь я использовал SHA-256 — это криптографическая хэш-функция используется для генерации фиксированного 256-битного (32 байта) хэша из произвольного входного сообщения. SHA-256 обеспечивает высокий уровень криптографической стойкости, что делает его устойчивым к

атакам типа "collision attack" (поиск двух различных входов, которые дают одинаковый хэш);

– подпись сообщения signature производится с помощью приватного ключа и хэш сообщения;
– verify для верификации подписи.

Класс ECDSA представлен на рисунке 13.

```
class ECDSA:
    def __init__(self, curve):
        self.curve = curve

    def generate_keys(self):
        private_key = SigningKey.generate(curve=self.curve)
        public_key = private_key.get_verifying_key()
        return private_key, public_key

    def sign(self, private_key, message):
        message_hash = sha256(message.encode()).digest()
        signature = private_key.sign(message_hash)
        return signature

    def verify(self, public_key, signature, message):
        message_hash = sha256(message.encode()).digest()
        try:
            return public_key.verify(signature, message_hash)
        except ecdsa.BadSignatureError:
            return False
```

Рисунок 13 – Реализация класса ECDSA

Данный класс содержит функции для генерации ключей, вычисления хэша сообщения и подписи, а также верификации.

В функции demo_ecdsa, которая отвечает за подтверждения подписей, я определил следующие вычисления:

Создал экземпляр кривой SECP256k1.

Создал объект ECDSA, который содержит эту кривую.

Составил сообщение для подписи.

Подписал сообщение и проверил подпись.

Функция демонстрации работы алгоритма demo_ecdsa представлена на рисунке 14.

```
def demo_ecdsa():

    curve = SECP256k1

    ecdsa = ECDSA(curve)

    private_key, public_key = ecdsa.generate_keys()

    message = "Hello, ECDSA!"

    signature = ecdsa.sign(private_key, message)

    is_valid = ecdsa.verify(public_key, signature, message)
```

Рисунок 14 – Функция демонстрации работы алгоритма ECDSA

Функция позволяет подтвердить подпись с использованием открытых и закрытых ключей.

Выводы итогам третьей главы: реализованы алгоритмы обмена ключами и цифровой подписи с использованием кривой SECP256k1 и стойкой хэш-функции SHA-256, также был задействован метод умножения в алгоритме ECDH.

Заключение

Выполненная работа по теме реализация криптографического алгоритма на эллиптических кривых является актуальной и востребованной в современном мире. Алгоритмы шифрования на эллиптических кривых нужны для обеспечения высокого уровня безопасности, производительности и совместимости, что делает их незаменимыми в различных областях, от интернет-протоколов и цифровой подписи до криптовалют и устройств IoT. С учетом растущих киберугроз и появления квантовых вычислений, дальнейшие исследования и разработки в этой области остаются критически важными для обеспечения безопасности данных и коммуникаций.

Поставленная цель достигнута. Задачи, поставленные в начале работы, были выполнены.

В процессе работы из теории эллиптических кривых были выделены ключевые элементы, лежащие в основе криптографии на их базе. Проанализированы основные криптографические алгоритмы, использующие эллиптические кривые. В частности, были рассмотрены алгоритмы обмена ключами, такие как ECDH, алгоритмы цифровой подписи, например, ECDSA, а также методы шифрования, такие как смешанный алгоритм ECIES.

Мною были разработаны классы для реализации методов внутри них.

Класс TFF для конечного поля с методами сложения, нахождения обратного элемента, скалярного умножения, отрицания, деления и сложения с обратным элементом, класс ECpoint для точки на эллиптической кривой, класс EC для эллиптической кривой, класс ECgroup для реализации методов сложения, скалярного умножения, нахождения обратной точки, проверки обратной точки, удвоения точки и сложения двух скалярных произведений.

Результатом работы является построенные классы с методами для работы на эллиптических кривых. Реализованы алгоритмы обмена ключами ECDH и цифровой подписи ECDSA.

Список используемой литературы и используемых источников

1. Бахаров Л. Е. Информационная безопасность и защита информации (разделы криптография и стеганография): практикум. Москва: МИСиС, 2019.
2. Борзунов С. В. Кургалин С. Д. Языки программирования. Python: решение сложных задач: учебное пособие для вузов. Санкт-Петербург: Лань, 2023.
3. Грибунин В. Г., Мартынов А. П., Николаев Д. Б., Фомченко В. Н. Криптография и безопасность цифровых систем: учебное пособие. ФГУП"РФЯЦ-ВНИИЭФ", 2017.
4. Данилов А. Н. Кротова Е. Л. Математические основы криптологии и криптографические методы и средства обеспечения информационной безопасности: учебное пособие для вузов. Пермь: Пермский государственный технический университет, 2008.
5. Демидов Д. Г. Швечкова О. Г., Москвитина О. А., Пылькин А. Н., Майков К. А., Смирнова Г. К. Защита информации с использованием механизмов электронной цифровой подписи: учебно-методическое пособие. Москва: Московсвкий государственный университет печати имени Ивана Федорова, 2014.
6. Донгак Ш. М. Криптография Часть II: Практикум. Москва: МИРЭА – Российский технологический университет, 2020.
7. Донгак Ш. М., Ушкова Н. Н. Криптография. Часть III. Москва: МИРЭА – Российский технологический университет, 2021.
8. Игнатьев Е. Б. Основы криптографии: учебное пособие. Иваново: Ивановский государственный энергетический университет имени В. И. Ленина, 2020. – 88с.

9. Ильиных С. Н. Алюшина С. Г., Калинкина Т. И., Янкевский А. В., Чернышев О. А. Введение в криптографическую защиту информации объектов: учебник для СПО. Москва: МТ КТУСИ, 2021.

10. Каширская Е. Н. Основы криптографического анализа [Электронный ресурс]: учебное пособие / Каширская Е. Н., Кушнир. А. П. – Москва: МИРЭА – Российский технологический университет, 2020.

11. Каширская Е. Н. Криптографический анализ и методы защиты информации [Электронный ресурс]: учебное пособие / Каширская Е. Н., Кушнир. А. П. – Москва: МИРЭА – Российский технологический университет, 2020.

12. Каширская Е. Н. Криптографичекие системы [Электронный ресурс]: Учебное пособие / Каширская Е. Н., Кушнир. А. П. – Москва: МИРЭА – Российский технологический университет, 2021.

13. Коржик В. И., Просихин В. П., Яковлев В. А. Основы криптографии: учебное пособие. Санкт-Петербург: СПбГУТ, 2014.

14. Майстренко, Н. В. Основы теории информации и криптографии: учебное пособие / Н. В. Майстренко, А. В. Майстренко. — Тамбов: ТГТУ, 2018.

15. Мартынов Л. М. Алгебра и теория числе для криптографии: учебное пособие для вузов. Санкт-Петербург: Лань, 2024.

16. Панкратова, И. А. Булевы функции в криптографии: учебное пособие / И. А. Панкратова. — Санкт-Петербург: Лань, 2022.

17. Пономарчук, Ю. В. Основы анализа шифров классической криптографии: учебное пособие / Ю. В. Пономарчук. — Хабаровск: ДВГУПС, 2019.

18. Рацеев С. М. Теоретико-числовые методы в криптографии. Часть 1: учебное пособие для вузов. Ульяновск.: УлГУ, 2020.

19. Сергеева О. А. Основы криптографии: Учебно-методическое пособие / О. А. Сергеева, А. С. Кутовая. — Кемерово: КемГУ, 2024.

20. Финогенов А. А. Введение в криптографию: Учебно-методическое пособие / А. А. Финогенов. — Ханты-Мансийск: ЮГУ, 2024.

21. Aumasson, J-P. Serious Cryptography: a Practical Introduction to Modern Encryption / J-P. Aumasson No Starch Press. - 2018.

22. Bancila, M. Modern C++ Programming Cookbook / M. Bancila Packt Publishing. - 2017.

23. Easttom W. Modern Cryptography / W. Easttom Applied Mathematics for Encryption and Information Security. - Springer. - 2021.

24. Neapolitan, R. E. Foundations of Algorithms. / R. E. Neapolitan Jones & Bartlett Learning. - 2015.

25. Shemanske T. R. Modern Cryptography and Elliptic Curves: a Beginners Guide / T. R. Shemanske American Mathematical Society. - 2017.

Приложение А

Листинг программы

Листинг 1 – реализация программы

```
from sympy import GF, symbols, poly
from ecdsa.ellipticcurve import CurveFp, Point
from ecdsa import curves, SigningKey, VerifyingKey, BadSignatureError,
SECP256k1
from hashlib import sha256

class TFF:
    def __init__(self, p, m):
        self.p = p
        self.m = m
        self.Fp = GF(p)
        self.x = symbols('x')
        self.irred = poly(self.x**m + self.x + 1, modulus=p)

    def add(self, a, b):
        return (a + b) % self.p

    def neg(self, a):
        return (-a) % self.p

    def inv(self, a):
        return pow(a, -1, self.p)

    def addInv(self, a, b):
```

Продолжение листинга 1

```
            return (a - b) % self.p

        def div(self, a, b):
            return (a * self.inv(b)) % self.p

        def scalarMult(self, a, k):
            return (a * k) % self.p

class ECpoint:
        def __init__(self, x, y):
            self.x = x
            self.y = y

        def __str__(self):
            return f"ECpoint({self.x}, {self.y})"

class EC:
        def __init__(self, a, b, p):
            self.a = a
            self.b = b
            self.p = p
            self.curve = CurveFp(p, a, b)

        def is_on_curve(self, pt):
            return self.curve.contains_point(pt.x, pt.y)
```

Продолжение листинга 1

```
class ECgroup:
    def __init__(self, curve, generator, order):
        self.curve = curve
        self.generator = generator
        self.order = order

    def add(self, pt1, pt2):
        return pt1 + pt2

    def scalar_mult(self, k, pt):
        return k * pt

    def inv(self, pt):
        return Point(self.curve, pt.x(), -pt.y() % self.curve.p())

    def isInv(self, pt1, pt2):
        return self.inv(pt1) == pt2

    def Double(self, pt):
        return pt + pt

    def compute(self, k1, pt1, k2, pt2):
        return (k1 * pt1) + (k2 * pt2)
```

Продолжение листинга 1

```
def ECDH(ecgroup, O_a, O_b):

    a_priv_key = SigningKey.generate(curve=SECP256k1)
    b_priv_key = SigningKey.generate(curve=SECP256k1)
    O_a = a_priv_key.verifying_key.pubkey.point
    O_b = b_priv_key.verifying_key.pubkey.point

S_a = ecgroup.scalar_mult(a_priv_key.privkey.secret_multiplier, O_b)
S_b = ecgroup.scalar_mult(b_priv_key.privkey.secret_multiplier, O_a)

assert S_a == S_b

return S_a

fld = TFF(7, 3)

ec = EC(2, 3, 7)

def find_point_on_curve(curve):
    for x in range(curve.p):
    for y in range(curve.p):
    if curve.curve.contains_point(x, y):
        return ECpoint(x, y)
    return None

pt1 = find_point_on_curve(ec)
```

Продолжение листинга 1

```
    pt2 = find_point_on_curve(ec)

    generator = Point(ec.curve, pt1.x, pt1.y)
    order = generator.order()

    ecgroup = ECgroup(ec.curve, generator, order)
    pt_sum = ecgroup.add(generator, generator)
    pt_scalar_mult = ecgroup.scalar_mult(3, generator)
    pt1_double = ecgroup.Double(generator)
    pt1_inv = ecgroup.inv(generator)

    if ecgroup.isInv(generator, generator):
        print("Точка Q является обратной к точке P")
    else:
        print("Точка Q не является обратной к точке P")

    shared_secret = ECDH(ecgroup, pt1, pt2)

class ECDSA:
    def __init__(self, curve):
        self.curve = curve

    def generate_keys(self):
        private_key = SigningKey.generate(curve=self.curve)
        public_key = private_key.get_verifying_key()
        return private_key, public_key
```

Продолжение листинга 1

```
    def sign(self, private_key, message):
        message_hash = sha256(message.encode()).digest()
        signature = private_key.sign(message_hash)
    return signature


    def verify(self, public_key, signature, message):
        message_hash = sha256(message.encode()).digest()
    try:
        return public_key.verify(signature, message_hash)
        except ecdsa.BadSignatureError:
    return False

    def demo_ecdsa():

        curve = SECP256k1
        ecdsa = ECDSA(curve)
        private_key, public_key = ecdsa.generate_keys()

        message = "Hello, ECDSA!"

        signature = ecdsa.sign(private_key, message)

        is_valid = ecdsa.verify(public_key, signature, message)
if __name__ == "__main__":
   demo_ecdsa()
```