

СЕТЕВАЯ БЕЗОПАСНОСТЬ В СИСТЕМАХ ПЕРЕДАЧИ ДАННЫХ

УЧЕБНОЕ ПОСОБИЕ

2026

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования «Санкт-Петербургский государственный университет
телекоммуникаций им. проф. М. А. Бонч-Бруевича»
(СПбГУТ)

СЕТЕВАЯ БЕЗОПАСНОСТЬ В СИСТЕМАХ ПЕРЕДАЧИ ДАННЫХ

Учебное пособие

Электронное издание
локального распространения

СПб ГУТ)))

Санкт-Петербург
Наукоемкие технологии
2026

© Авторский коллектив, 2026
© Санкт-Петербургский государственный университет
телекоммуникаций им. проф. М. А. Бонч-Бруевича, 2026
ISBN 978-5-00271-188-8

УДК 621.391:004.056 (075.8)
ББК 32.84:16.8я73
С33

Авторы:

Д. С. Кукунин, А. А. Березкин, А. В. Бородко, А. В. Пачин, Р. В. Киричек

Рецензенты:

И. В. Карташевский, доктор технических наук, профессор кафедры
программной инженерии Поволжского государственного университета
телекоммуникаций и информатики;

А. В. Рабин, доктор технических наук, директор центра специальных проектов
Санкт-Петербургского государственного университета телекоммуникаций
им. проф. М. А. Бонч-Бруевича

С33 Сетевая безопасность в системах передачи данных [Электронный ресурс]: учебное пособие / Д. С. Кукунин, А. А. Березкин, А. В. Бородко, А. В. Пачин, Р. В. Киричек; СПбГУТ. – Электрон, текстовые дан. (1,8 Мб). – СПб.: Научные технологии, 2026. – 89 с. – 1 электрон., опт. диск (CD-ROM).

ISBN 978-5-00271-188-8

Рассмотрены основы сетевой безопасности. Приведены примеры реализации наиболее известных на сегодняшний день методов защиты информации в системах передачи данных. Представленные здесь алгоритмы имеют полное теоретическое описание, при этом каждый раздел сопровождается подробным примером и содержит контрольное задание, предназначенное для самостоятельного выполнения обучающимися.

Предназначено для студентов, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника» и 11.03.02 «Инфокоммуникационные технологии и системы связи».

*Утверждено редакционно-издательским советом СПбГУТ
в качестве учебного пособия
Протокол № 8 от 29.04.2025*

Текстовое электронное издание

Минимальные системные требования:

- процессор: Intel x86, x64, AMD x86, x64 не менее 1 ГГц;
- оперативная память RAM ОЗУ: не менее 512 Мбайт;
- свободное место на жестком диске (HDD): не менее 120 Мбайт;
- операционная система: Windows XP и выше;
- Adobe Acrobat Reader;
- дисковод CD-ROM;
- мышь.

УДК 621.391:004.056 (075.8)
ББК 32.84:16.8я73

ISBN 978-5-00271-188-8

© Авторский коллектив, 2026
© Санкт-Петербургский государственный университет
телекоммуникаций им. проф. М. А. Бонч-Бруевича, 2026

Учебное издание

Кукунин Дмитрий Сергеевич
Березкин Александр Александрович
Бородко Александр Владимирович
Пачин Андрей Владимирович
Киричек Руслан Валентинович

Сетевая безопасность в системах передачи данных

Учебное пособие

Электронное издание
локального распространения

Издается в авторской редакции
Публикуется из оригинал-макета, подготовленного авторами

Издательство «Наукоемкие технологии»
ООО «Корпорация «Интел Групп»
<https://publishing.intelgr.com>
E-mail: publishing@intelgr.com
Тел.: +7 (812) 945-50-63

Подписано к использованию 13.10.2026 г.
Объем издания – 1,8 Мб.
Комплектация издания – 1 CD.
Тираж 100 CD.

ISBN 978-5-00271-188-8



9 785002 711888 >

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1. ОСНОВЫ ТЕОРИИ ЗАЩИТЫ ИНФОРМАЦИИ	7
1.1. Определения и термины	7
1.2. Направления по защите информации	8
1.3. Конечные поля в криптографии	10
1.3.1. Функция Эйлера и первообразные элементы	10
1.3.2. Обратный элемент	13
1.3.3. Расширенный алгоритм Евклида	13
1.3.4. Дискретный логарифм	16
1.3.5. Расширенные поля	17
1.3.6. Операции над элементами расширенных полей	18
Задание на самостоятельную работу	20
1.4. Эллиптические кривые	22
1.4.1. Принципы построения эллиптических кривых	22
1.4.2. Точки эллиптической кривой	24
1.4.3. Удвоение точки эллиптической кривой	27
1.4.4. Скалярное умножение точки эллиптической кривой	28
1.4.5. Эллиптические кривые над конечным полем	29
1.4.6. Порядок группы точек эллиптической кривой	30
1.4.7. Циклические подгруппы точек эллиптической кривой	31
Задание на самостоятельную работу	33
2. ЦЕЛОСТНОСТЬ ДАННЫХ	35
2.1. Функции хеширования	35
2.2. Хеширование в OpenSSL	37
2.3. Концепция блокчейн	41
2.3.1. Транзакции в сети блокчейн	42
2.3.2. Дерево Меркла	44
2.3.3. Доказательство Меркла	47
Задание на самостоятельную работу	50
3. КРИПТОГРАФИЧЕСКИЕ СИСТЕМЫ	51
3.1. Шифр Виженера и его реализация.....	51
Задание на самостоятельную работу	54
3.2. Метод Диффи–Хеллмана	54
3.2.1. Схема Диффи–Хеллмана для двух абонентов	55
3.2.2. Схема Диффи–Хеллмана для множества абонентов	56
Задание на самостоятельную работу	57
3.3. Схема разделения секрета Шамира	58
3.3.1. Процедура разделения и восстановления секрета	59
3.3.2. Реализация схемы Шамира	61
Задание на самостоятельную работу	63
3.4. Схема Эль-Гамала	64
3.4.1. Цифровая подпись на основе схемы Эль-Гамала	64
3.4.2. Шифрование с открытым ключом по схеме Эль-Гамала	67
Задание на самостоятельную работу	69
3.5. Криптосистема RSA	70

3.5.1. Создание ключей RSA	70
3.5.2. Цифровая подпись на основе RSA	72
3.5.3. Шифрование на основе RSA	73
Задание на самостоятельную работу	74
3.6. Схема ECDSA	75
3.6.1. Цифровая подпись на основе схемы ECDSA	76
3.6.2. Поддержка ECDSA в OpenSSL	80
Задание на самостоятельную работу	84
КОНТРОЛЬНЫЕ ВОПРОСЫ	86
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	88

ВВЕДЕНИЕ

Данное пособие затрагивает математические основы современных методов обеспечения сетевой безопасности и дает возможность обучающимся самостоятельно выполнить ряд типовых заданий для лучшего понимания специфики дисциплины «Защита информации в системах передачи данных», а также дисциплин, связанных с ней логическим образом.

Индивидуальные задания по каждому приведенному здесь алгоритму построены так, что могут быть выполнены вручную с использованием подручных средств, в частности простых калькуляторов или табличных процессоров. Но это совершенно не исключает использования студентами различных языков программирования, которые могут оказаться эффективным инструментом реализации того или иного метода обработки информации и вместе с тем повысить их навыки владения современными технологиями.

Хотелось бы высказать отдельную благодарность И. Р. Сергиенко, который, выполняя свою дипломную работу, любезно предоставил материалы для организации практических занятий по ряду направлений, рассмотренных в данном пособии.

1. ОСНОВЫ ТЕОРИИ ЗАЩИТЫ ИНФОРМАЦИИ

Данный раздел содержит информацию прежде всего теоретического характера. Сначала читатель знакомится с основными понятиями и терминами теории криптографии, затем открывает для себя теорию конечных полей, в частности функцию Эйлера и малую теорему Ферма. Это позволяет в итоге понять смысл нахождения первообразного корня, который используется во многих криптографических системах, в том числе в схеме Диффи – Хеллмана и Эль-Гамала.

Также здесь присутствуют отсылки к популярным на сегодняшний день криптографическим алгоритмам, которые используют математический аппарат, основанный на эллиптических кривых.

1.1. Определения и термины

Федеральный закон РФ от 27.07.2006 г. № 149-ФЗ «Об информации, информатизации и защите информации» определяет следующие термины в сфере телекоммуникационных систем и информационной безопасности.

Защита информации – это защита (сохранение целостности и конфиденциальности) информации от несанкционированного доступа.

Целостность информации – это сохранение ее в первоначальном виде.

Конфиденциальность – означает, что информацию должен получить только тот, кому она и предназначалась.

Шифрование – обратимое преобразование информации в целях сокрытия от сторонних (неавторизованных) лиц с предоставлением в это же время авторизованным легитимным пользователям доступа к ней. Шифрование обеспечивает конфиденциальность передаваемой информации.

Криптография – наука о методах обеспечения конфиденциальности и аутентичности (целостности и подлинности авторства, а также невозможности отказа от авторства) информации.

Стеганография – это наука о скрытой передаче информации путем сохранения в тайне самого факта передачи.

Криптоаналитик – это нелегальный пользователь (злоумышленник), который стремится получить несанкционированный доступ к информации.

Криптоустойчивость криптосистемы определяется числом стандартных операций, которые криптоаналитик должен выполнить для достижения цели, не зная при этом ключей шифрования.

Двоичная (битовая) криптоустойчивость (сложность) определяется количеством двоичных операций для достижения цели криптоаналитика.

Криптоатака – попытка криптоаналитика получить информацию.

Пассивный криптоаналитик может осуществлять подслушивание и (или) перехват зашифрованного сообщения, не нарушая при этом процесс обмена зашифрованными сообщениями между отправителем и получателем.

Активный криптоаналитик может подслушивать, перехватывать и преобразовывать сообщения по своему усмотрению: задерживать, искажать, имитировать создание новых сообщений, устраивать обрыв связи и т. п. В этом случае одновременно нарушается и целостность передаваемого сообщения и конфиденциальность.

Взлом или вскрытие криптосистемы – удачная попытка получения информации нелегальным пользователем.

Криптоанализ – метод вскрытия криптосистемы.

Временная криптоустойчивость – это время (от нескольких месяцев до года и более), которое может затратить криптоаналитик для взлома криптосистемы.

1.2. Направления по защите информации

Имеет смысл выделить основные направления, по которым развиваются методы, обеспечивающие информационную безопасность современных систем обработки и передачи данных.

Аутентификация сторон является важнейшим условием безопасного обмена информацией между двумя и более узлами, будь то соединение точка-точка или сеть общего доступа. Результатом аутентификации должно

стать доверие к отправителю или получателю данных. Отсутствие доверия между сторонами ставит под вопрос смысл их дальнейшего обмена информацией. Аутентификация может быть односторонней или двусторонней, в зависимости от решаемых системой задач.

Аутентификация строится на процедуре проверки подлинности, которая реализуется, как правило, на основе некоторых индивидуальных данных. В качестве таких данных могут выступать, например, биометрические признаки и классические подписи, а также цифровые подписи, процедуры использования которых в современных системах передачи данных легко автоматизируются, обеспечивая тем самым работу с цифровыми сертификатами при организации TLS- и VPN-соединений.

Целостность данных является неотъемлемой частью безопасного обмена данными. Несмотря на доверие сторон, передаваемая информация также требует подтверждения доверия к ее содержанию. Основным решением здесь являются, как правило, хеш-функции, а также имитовставки, которые позволяют принимающей стороне убедиться в том, что принятый блок данных не был подвергнут преднамеренному искажению в процессе своей передачи.

Необходимо отметить, что аутентификация сторон и целостность данных являются обязательными требованиями к организации безопасного соединения по открытому каналу связи. Это касается, в частности, наиболее распространенного на сегодняшний день открытого программного продукта OpenVPN, который позволяет строить собственные VPN соединения, которые обеспечивают безопасный обмен данными с аутентификацией и защитой целостности данных.

Скрытие информации, как опциональная составляющая стандартного OpenVPN-соединения, является важным компонентом защиты данных от несанкционированного доступа.

Само по себе скрытие информации можно разделить на два основных направления: шифрование, которое обеспечивает защиту видимой криптоаналитиком информации от прочтения, и стеганографию, которая призвана скрыть сам факт ее передачи.

Несмотря на свою опциональность в рамках такого популярного продукта, как OpenVPN, *шифрование* с давних пор остается важнейшим направлением по поддержанию безопасности обмена информацией в открытых каналах передачи данных.

Шифрование с организационной точки зрения можно разделить на *симметричное* и *асимметричное*. Симметричное шифрование предполагает наличие общего секретного ключа, который используют передающая и принимающая стороны. Асимметричное шифрование построено на использовании двух ключей: публичного и секретного.

Необходимо сразу отметить, что ключи алгоритмов асимметричного шифрования так или иначе связаны между собой математически. Это можно считать их уязвимостью. Поэтому на практике ключи асимметричных криптосистем имеют довольно большую длину, что в совокупности со сложностью математических операций алгоритмов сильно замедляет процесс самой передачи информации.

Симметричные алгоритмы шифрования, напротив, просты в использовании и являются более криптостойкими. Вместе с тем постоянное использование одних и тех же общих ключей передающей и приемной сторонами неизбежно приводит к накоплению статистики у криптоаналитика. Поэтому важным требованием симметричного шифрования является регулярное обновление секретного ключа.

Приведенные в данном пособии методы защиты информации используют общие математические аппараты, реализация которых в свою очередь затрагивает не только теорию множеств, но и другие разделы алгебры. Имеет смысл изучить эти вопросы более подробно.

1.3. Конечные поля в криптографии

Приведенный далее материал предполагает то, что читатели хорошо ориентируются в теории множеств и знакомы с такими понятиями, как «группа», «кольцо», «поле» и помнят сопровождающие их аксиомы. Поэтому целесообразно сразу перейти к математическим аппаратам, непосредственно связанным с теорией криптографии.

1.3.1. Функция Эйлера и первообразные элементы

Функция Эйлера $\varphi(n)$ играет важную роль при вычислении так называемых первообразных корней или элементов мультипликативной группы кольца вычетов по $\text{mod } n$. Значение $\varphi(n)$ равно числу целых положительных чисел, меньших n и взаимно простых с ним [1].

Согласно теореме Эйлера, для взаимно простых a и n будет справедливо тождество:

$$a^{\varphi(n)} \equiv 1 \pmod{n}. \quad (1)$$

Для того чтобы число a считалось первообразным элементом мультипликативной группы кольца вычетов по $\text{mod } n$, необходимо выполнение условия [2]:

$$a^l \not\equiv 1 \pmod{n}, \quad 1 \leq l < \varphi(n). \quad (2)$$

Смысл (2) заключается в том, что, последовательное возведение элемента a во все степени от 1 до $\varphi(n)$ позволяет полностью сформировать ту самую мультипликативную группу кольца вычетов по $\text{mod } n$. При этом равенство $l = \varphi(n)$ приведет к выполнению (1). То есть единичный элемент группы не должен быть получен раньше всех остальных.

Очевидно, что при $n = p$, где p – простое число, значение функции Эйлера составит $\varphi(n = p) = p - 1$, так как n взаимно простое со всеми меньшими его числами. В этом случае (1) можно записать в виде тождества:

$$a^{p-1} \equiv 1 \pmod{p}, \quad (3)$$

которое является основой малой теоремы Ферма.

Тождество (3) позволяет рассматривать мультипликативную группу кольца вычетов по $\text{mod } n$ как простое конечное поле Галуа $\text{GF}(p)$ с характеристикой p , где p – простое число [3].

Приведем пример простого поля $\text{GF}(5)$, где первообразным элементом является число 2, которое удовлетворяет условию (2) и при последовательном возведении в степени формирует остальные элементы (от 1 до $p - 1 = 4$), которые, в свою очередь, циклически повторяют полный период поля:

$$\begin{aligned} 2^1 &\equiv 2 \pmod{5}, \\ 2^2 &\equiv 4 \pmod{5}, \\ 2^3 &= 8 \equiv 3 \pmod{5}, \\ 2^4 &= 16 \equiv 1 \pmod{5}, \\ 2^5 &= 32 \equiv 2 \pmod{5}, \\ 2^6 &= 64 \equiv 4 \pmod{5}, \\ 2^7 &= 128 \equiv 3 \pmod{5}, \\ 2^8 &= 256 \equiv 1 \pmod{5}, \\ 2^9 &= 512 \equiv 2 \pmod{5}, \\ 2^{10} &= 1024 \equiv 4 \pmod{5}, \\ &\dots \end{aligned}$$

Проверка условия (2) для элемента 2 в поле $\text{GF}(7)$ дает отрицательный результат, так как процесс заикливается раньше положенного срока, не позволяя сформировать все элементы (от 1 до $p - 1 = 6$). Здесь первообразным элементом является число 3:

$$\begin{array}{ll} 2^1 \equiv 2 \pmod{7}, & 2^6 = 64 \equiv 1 \pmod{7}, \\ 2^2 \equiv 4 \pmod{7}, & 2^7 = 128 \equiv 2 \pmod{7}, \\ 2^3 = 8 \equiv 1 \pmod{7}, & 2^8 = 256 \equiv 4 \pmod{7}, \\ 2^4 = 16 \equiv 2 \pmod{7}, & 2^9 = 512 \equiv 1 \pmod{7}, \\ 2^5 = 32 \equiv 4 \pmod{7}, & 2^{10} = 1024 \equiv 2 \pmod{7}, \\ & \dots \end{array}$$

$$\begin{array}{ll}
3^1 \equiv 3 \pmod{7}, & 3^6 = 729 \equiv 1 \pmod{7}, \\
3^2 = 9 \equiv 2 \pmod{7}, & 3^7 = 2187 \equiv 3 \pmod{7}, \\
3^3 = 27 \equiv 6 \pmod{7}, & 3^8 = 6561 \equiv 2 \pmod{7}, \\
3^4 = 81 \equiv 4 \pmod{7}, & 3^9 = 19683 \equiv 6 \pmod{7}, \\
3^5 = 243 \equiv 5 \pmod{7}, & 3^{10} = 59049 \equiv 4 \pmod{7}, \\
& \dots
\end{array}$$

На основании вышесказанного можно сделать ряд выводов.

1. Циклическая мультипликативная группа кольца вычетов по $\text{mod } n$ включает в себя $\varphi(n)$ элементов, которые не имеют общих делителей с n .

2. Среди этих $\varphi(n)$ элементов могут быть первообразные элементы, которые удовлетворяют условию (2).

3. Если данная группа является полем, то среди его элементов должен быть хотя бы один первообразный корень.

Следует еще раз подчеркнуть важное свойство первообразного элемента, которое может оказаться полезным для ускорения процесса его нахождения. Пусть $\varphi(p) = c$, при этом число c имеет набор делителей q_i , где $i = 1, 2, \dots, l$. Тогда для первообразного элемента a достаточно выполнение условий:

$$a^{\frac{c}{q_i}} \not\equiv 1 \pmod{p}, \quad i = 1, 2, \dots, l. \quad (4)$$

Приведем пример использования данного свойства для поля $\text{GF}(7)$. Значение функции Эйлера $c = \varphi(7) = 7 - 1 = 6$. Число $c = 6$ имеет два простых множителя: $q_1 = 2$ и $q_2 = 3$, которые являются его делителями: $c = 2 \cdot 3$.

Первообразный элемент в соответствии с (4) должен удовлетворять обоим условиям:

$$a^{\left(\frac{6}{q_1=2}\right)=3} \not\equiv 1 \pmod{7}; \quad a^{\left(\frac{6}{q_2=3}\right)=2} \not\equiv 1 \pmod{7}.$$

Действительно, последовательная проверка элементов поля $\text{GF}(7)$ показывает, что число 2 не является первообразным элементом, так как сразу же не проходит проверку с делителем q_1 : $2^3 = 8 \equiv 1 \pmod{7}$.

Число 3 проходит проверку полностью и удовлетворяет обоим условиям с делителями q_1 и q_2 :

$$3^3 = 27 \equiv 6 \pmod{7}; \quad 3^2 = 9 \equiv 2 \pmod{7}.$$

Таким образом, можно подтвердить результат, приведенный выше, в том числе равенство (3):

$$3^6 = 729 \equiv 1 \pmod{7}.$$

1.3.2. Обратный элемент

Для $GF(p)$ с первообразным элементом a характерно наличие элемента $a^0 = 1$, который выполняет роль мультипликативной единицы и позволяет утверждать, что каждый элемент поля a^i имеет обратный элемент по умножению $b^j = a^{-i}$, принадлежащий этому же полю [2]. То есть справедливо равенство:

$$a^i b^j = a^i a^{-i} \equiv 1 \pmod{p}. \quad (5)$$

Обратные элементы b^j для a^i , отвечающие условию (5), хорошо просматриваются в таблицах истинности по умножению. Приведем пример таких таблиц для полей $GF(5)$ и $GF(7)$, где единичные элементы будут стоять на пересечении строк и столбцов, содержащих обратные друг для друга элементы:

GF(5)					GF(7)						
$\times$	1	2	3	4	$\times$	1	2	3	4	5	6
1	1	2	3	4	1	1	2	3	4	5	6
2	2	4	1	3	2	2	4	6	1	3	5
3	3	1	4	2	3	3	6	2	5	1	4
4	4	3	2	1	4	4	1	5	2	6	3
					5	5	3	1	6	4	2
					6	6	5	4	3	2	1

Как видно из таблиц истинности, возможны случаи, когда сам элемент будет являться для себя обратным. В поле $GF(5)$ это числа 1 и 4, в поле $GF(7)$ это числа 1 и 6.

1.3.3. Расширенный алгоритм Евклида

Для нахождения обратного элемента в группе большого порядка может быть использован расширенный алгоритм Евклида, реализация которого сводится к решению следующего уравнения:

$$\alpha x + \beta y = c, \quad (6)$$

где $c = \text{НОД}(\alpha, \beta)$ – наибольший общий делитель для известных целых чисел α и β , при этом $\alpha > \beta$. Требуется найти значения x и y .

Реализация алгоритма сводится к набору шагов, первым из которых является предварительная инициализация переменных в уравнении (6).

Инициализация:

$$\begin{aligned} A &= \alpha, & B &= \beta, \\ A_0 &= 1, & B_0 &= 0, \\ A_1 &= 0, & B_1 &= 1. \end{aligned}$$

Шаг 1:

$$\begin{aligned} A &= qB + r, \text{ где } q - \text{частное, } r - \text{остаток,} \\ t &= A_0, & t &= B_0, \\ A_0 &= A_1, & B_0 &= B_1, \\ A_1 &= t - A_1q, & B_1 &= t - B_1q. \end{aligned}$$

Шаг 2

Выполняется проверка данных, полученных на шаге 1 (рис. 1).

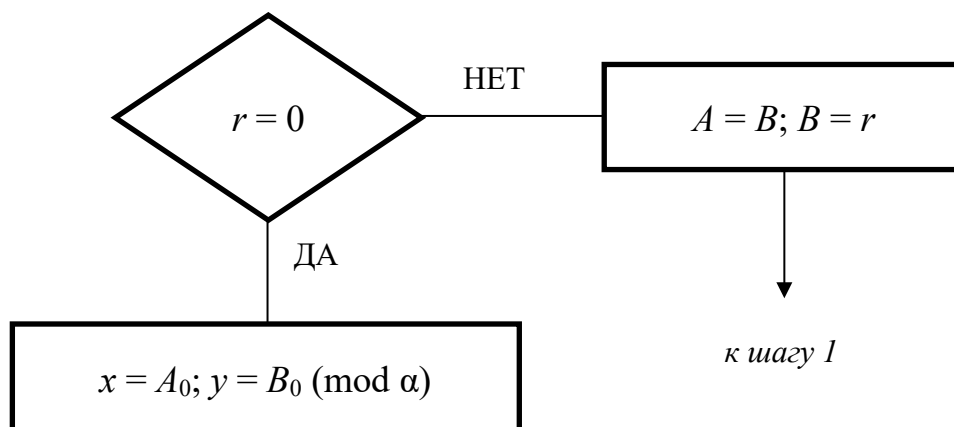


Рис. 1. Фрагмент алгоритма на шаге 2

Результат проверки условия на шаге 2 определяет необходимость следующей итерации алгоритма, т. е. возвращение к шагу 1.

Расширенный алгоритм Евклида может быть использован во многих криптосистемах, в частности в схеме Эль-Гамала и RSA.

Приведем пример нахождения обратного элемента d для элемента e в группе кольца вычетов по $\text{mod } N$. В данном случае будет справедливо тождество:

$$de \equiv 1(\text{mod } N),$$

которое может быть записано в форме (6):

$$Nk + ed \equiv 1(\text{mod } N),$$

где N и e известны, требуется найти k и d .

Таким образом, можно сделать вывод, что $\alpha = N$, $\beta = e$, $k = x$, $d = y$.

Пусть для примера $N = 288$, $e = 11$. Тогда получим следующее выражение, связывающее обратные элементы группы:

$$d \cdot 11 \equiv 1 \pmod{288},$$

которое запишем в виде (6):

$$288k + 11d \equiv 1 \pmod{288}.$$

Применим расширенный алгоритм Евклида для нахождения неизвестных параметров x и y в уравнении (6).

Сначала произведем инициализацию:

$$\begin{aligned} A &= \alpha = N, & B &= \beta = 11, \\ A_0 &= 1, & B_0 &= 0, \\ A_1 &= 0, & B_1 &= 1. \end{aligned}$$

Шаг 1:

$$\begin{aligned} A &= qB + r; 288 = q \cdot 11 + r; \\ q &= 26; r = 2; \\ t &= A_0 = 1; & t &= B_0 = 0; \\ A_0 &= A_1 = 0; & B_0 &= B_1 = 1; \\ A_1 &= t - A_1q = 1 - 0 \cdot 26 = 1; & B_1 &= t - B_1q = 0 - 1 \cdot 26 = -26. \end{aligned}$$

Шаг 2:

$$\begin{aligned} r &\neq 0; \\ A &= B = 11; B = r = 2; \\ &\text{переход к шагу 1.} \end{aligned}$$

Шаг 1:

$$\begin{aligned} A &= qB + r; 11 = q \cdot 2 + r; \\ q &= 5; r = 1; \\ t &= A_0 = 0; & t &= B_0 = 1; \\ A_0 &= A_1 = 1; & B_0 &= B_1 = -26; \\ A_1 &= t - A_1q = 0 - 1 \cdot 5 = -5; & B_1 &= t - B_1q = 1 - (-26) \cdot 5 = 131. \end{aligned}$$

Шаг 2:

$$\begin{aligned} r &\neq 0; \\ A &= B = 2; B = r = 1; \\ &\text{переход к шагу 1.} \end{aligned}$$

Шаг 1:

$$A = qB + r; 2 = q \cdot 1 + r;$$

$$q = 2; r = 0;$$

$$t = A_0 = 1;$$

$$t = B_0 = -26;$$

$$A_0 = A_1 = -5;$$

$$B_0 = B_1 = 131.$$

Шаг 2:

$$r = 0;$$

$$k = x = A_0 = -5 = 283 \pmod{288};$$

$$d = y = B_0 = 131 \pmod{288}.$$

Таким образом, получим справедливое равенство:

$$288 \cdot 283 + 11 \cdot 131 \equiv 1 \pmod{288},$$

которое легко упрощается до выражения:

$$11 \cdot 131 \equiv 1 \pmod{288},$$

где элементы групп 11 и 131 являются взаимно-обратными.

1.3.4. Дискретный логарифм

Произвольный элемент поля $GF(p)$ имеет две формы представления, которые связаны через функцию дискретного логарифма [4–8]:

$$z = \log_a a^z \pmod{p}, \quad (7)$$

где a – первообразный элемент поля; $a^z \pmod{p}$ – число, соответствующее произвольному элементу поля; z – показатель степени a , определяющий этот произвольный элемент поля.

Из равенства (7) видно, что для нулевого элемента поля дискретного логарифма не существует, поскольку элемент 0 невозможно получить возведением первообразного элемента a в какую бы то ни было степень.

Таким образом, любой ненулевой элемент поля $GF(p)$ может быть представлен числом от 1 до $(p - 1)$ или степенью первообразного элемента этого поля.

Проблема вычисления дискретного логарифма в конечном поле является одной из основных для криптосистем с открытым ключом. На сегодняшний день она представляется достаточно сложной и не имеет эффективных универсальных методов решения, однако существуют алгоритмы

и подходы, способные ускорить процесс вычисления дискретного логарифма в конечном поле [9–13].

Так или иначе, процедура дискретного логарифмирования положена в основу алгоритма Диффи – Хеллмана, где применяется для формирования общего секретного ключа, а также используется в криптосистеме Эль-Гамала.

1.3.5. Расширенные поля

Поля Галуа вида $GF(p^k)$ с натуральным целым показателем $k > 1$ принято называть расширенными полями [2]. Элементами таких полей являются векторы $(a_0 \ a_1 \ a_2 \ \dots \ a_{k-1})$ длины k или соответствующие им многочлены $a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1}$ с коэффициентами a_i , принадлежащими простому полю $GF(p)$.

Расширенное поле Галуа $GF(p^k)$ может быть построено на основе характеристического неприводимого многочлена $P(x)$, который удовлетворяет тождеству:

$$x^{p^k-1} - 1 \equiv 0 \pmod{P(x)}.$$

Таким образом, $P(x)$ делит двучлен $x^n - 1$, где $n = p^k - 1$, без остатка. Вместе с тем $P(x)$ является примитивным, так как его корень будет первообразным элементом данного поля и образует все ненулевые элементы путем последовательного возведения в степени от 1 до $p^k - 1$.

В случае $p = 2$ речь идет о двоичных расширенных полях $GF(2^k)$, элементами которых будут двоичные векторы длины k . Такие поля решают важные задачи в области помехоустойчивого кодирования, прежде всего используются для построения циклических кодов, способных обнаруживать и исправлять ошибки, например кодов БЧХ и Рида – Соломона [2].

Приведем пример двоичного расширенного поля $GF(2^4)$ с первообразным элементом ε , построенного на основе полинома $P(x) = x^4 + x + 1$ (табл. 1).

Таблица 1

Поле $GF(2^4)$ на основе $P(x) = x^4 + x + 1$

Степень элемента ε^j	Вектор $(a_0 \ a_1 \ a_2 \ a_3)$
$\varepsilon^0 = 1$	1000
ε	0100
ε^2	0010
ε^3	0001
ε^4	1100

Степень элемента ε^j	Вектор $(a_0 a_1 a_2 a_3)$
ε^5	0110
ε^6	0011
ε^7	1101
ε^8	1010
ε^9	0101
ε^{10}	1110
ε^{11}	0111
ε^{12}	1111
ε^{13}	1011
ε^{14}	1001

Как видно из табл. 1, все элементы расширенного поля, по аналогии с простым полем, выражаются через степень первообразного элемента. При этом их векторное или полиномиальное представление, приведенное по mod $P(x)$, можно считать аналогом числу, приведенному по mod p в простом поле. Таким образом, для расширенного поля характерны те же операции сложения и умножения, с учетом того, что результат этих операций приводится по mod $P(x)$.

1.3.6. Операции над элементами расширенных полей

Пусть элементы ε^i на ε^j , принадлежащие некоторому полю $\text{GF}(2^k)$, выражены через левый степенной базис поля:

$$\begin{aligned}\varepsilon^i &= a_0 + a_1\varepsilon + a_2\varepsilon^2 + \dots + a_{k-1}\varepsilon^{k-1}, \\ \varepsilon^j &= b_0 + b_1\varepsilon + b_2\varepsilon^2 + \dots + b_{k-1}\varepsilon^{k-1}.\end{aligned}\tag{8}$$

Тогда их умножение равнозначно произведению многочленов с коэффициентами $a_0, a_1, a_2, \dots, a_{k-1}$ и $b_0, b_1, b_2, \dots, b_{k-1}$ соответственно:

$$\begin{aligned}\varepsilon^m = \varepsilon^i \varepsilon^j &= (a_0 b_0) + (a_0 b_1 + a_1 b_0)\varepsilon + (a_0 b_2 + a_1 b_1 + a_2 b_0)\varepsilon^2 + \dots \\ &\dots + a_{k-1} b_{k-1} \varepsilon^{2k-2} = c_0 + c_1 \varepsilon + c_2 \varepsilon^2 + \dots + c_{k-1} \varepsilon^{k-1}.\end{aligned}\tag{9}$$

Для двоичного расширенного поля коэффициенты в (9) определяются следующим образом:

$$c_i \equiv \sum_{j=0}^i a_j b_{i-j} \pmod{2}.$$

Произведение элементов поля $GF(2^k)$, представленных векторами или полиномами (8), может быть реализовано на основании схемы умножения (рис. 2) [2].

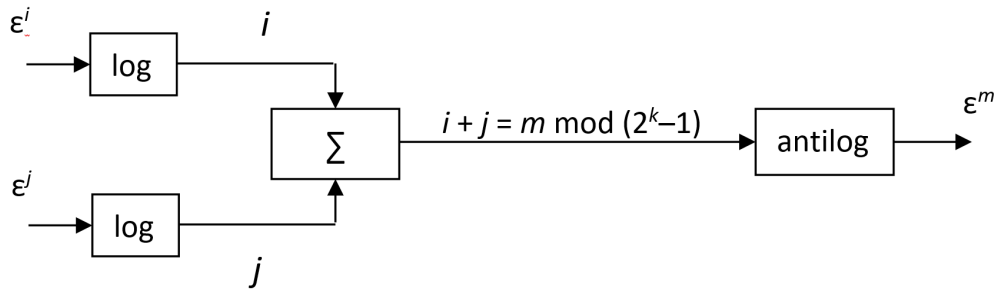


Рис. 2. Алгоритм умножения элементов поля $GF(2^k)$

Процесс вычисления произведения ε^i и ε^j начинается с определения их дискретных логарифмов (7). Затем вычисляется сумма показателей их степеней i и j , при этом результат сложения приводится по $\text{mod } (2^k - 1)$. После этого параметр m определяет элемент ε^m и возвращает результат умножения в векторную форму элемента поля.

Сложение элементов расширенного поля в виде многочленов (8) сводится к суммированию их коэффициентов при одинаковых степенях и приведению суммы по $\text{mod } p$:

$$\varepsilon^m = \varepsilon^i + \varepsilon^j = (a_0 + b_0) + (a_1 + b_1)x + (a_2 + b_2)x^2 + \dots + (a_{k-1} + b_{k-1})x^{k-1}. \quad (10)$$

Равенство (10) может быть записано в виде:

$$\varepsilon^m = c_0 + c_1x + c_2x^2 + \dots + c_{k-1}x^{k-1}. \quad (11)$$

При этом (10) и (11) в случае двоичного расширенного поля связаны тождеством [2]:

$$(a_i + b_i) \equiv c_i \pmod{2}. \quad (12)$$

Сложение элементов ε^i и ε^j , представленных в степенном виде, реализуется схемой (рис. 3) [2].

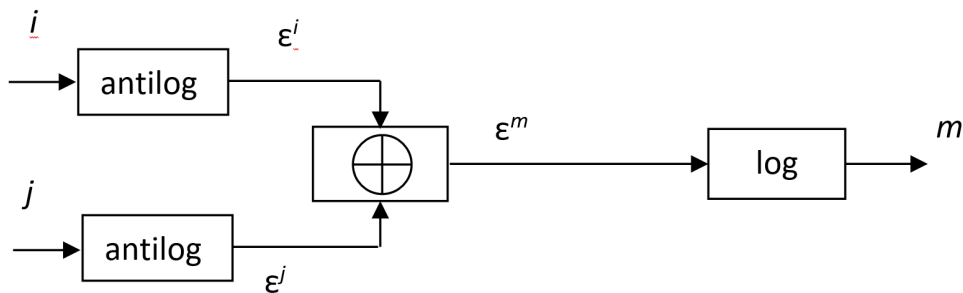


Рис. 3. Алгоритм сложения элементов поля $GF(2^k)$

Векторы ε^i и ε^j определяются при помощи антилогарифмирования. Эти векторы суммируются по правилу (12). Далее для ε^m определяется дискретный логарифм, т. е. вычисляется искомый показатель степени элемента поля.

Что касается нахождения обратного элемента для ε^j в двоичном расширенном поле, то здесь можно воспользоваться свойством $\text{GF}(2^k)$, ненулевые элементы которого образуют циклическую мультипликативную группу порядка $2^k - 1$.

Для нахождения ε^{-j} элемент ε^j достаточно возвести в степень $(2^k - 2)$ [2]:

$$\varepsilon^{-j} = (\varepsilon^j)^{2^k - 2} = \varepsilon^{j(2^k - 2)}. \quad (13)$$

Из равенства (13) следует, что задача обращения ε^j сводится к его возведению в степень $j(2^k - 2)$.

Как было показано выше, все операции над элементами расширенного поля аналогичны операциям в простом поле, за исключением того, что вместо обычных чисел расширенное поле оперирует векторами.

Если говорить о применимости двоичных расширенных полей в области криптографии, то следует упомянуть самый популярный на сегодняшний день симметричный алгоритм шифрования AES, который в явном виде использует поле $\text{GF}(2^8)$.

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

1) вычислить при помощи расширенного алгоритма Евклида обратный элемент для элемента e в группе кольца вычетов по $\text{mod } N$ из табл. 2;

2) для многочлена $P_1(x)$ из табл. 2 определить, является ли он неприводимым. Достаточно проверить следующее свойство:

$$x^n - 1 \equiv 0 [\text{mod } P(x)],$$

выполнение которого говорит о том, что двучлен $x^n - 1$, где $n = 2^k - 1$, делится на $P_1(x)$ без остатка;

3) для многочлена $P_2(x)$ из табл. 2 определить, является ли он примитивным. Достаточно организовать на основе $P_2(x)$ процедуру построения двоичного расширенного поля Галуа. Если корень $P_2(x)$ не является первообразным элементом поля, то его последовательное возведение в степень от 1 до $2^k - 1$ не позволит получить полный набор всех ненулевых элементов данного поля;

4) используя поле $GF(2^4)$, элементы которого представлены в табл. 1, вычислить результат выражения C из табл. 2.

Таблица 2

Варианты заданий согласно номеру в журнале обучающихся

№ вар.	e	N	$P_1(x)$	$P_2(x)$	C
1	155	1632	$x^4 + x^3 + 1$	$x^5 + x^2 + 1$	$\left(\frac{\varepsilon^6 + \varepsilon^7}{\varepsilon^3}\right)^{56}$
2	19	1728	$x^5 + x^2 + 1$	$x^5 + x^3 + 1$	$(\varepsilon^{12} + \varepsilon^{10} + \varepsilon^{-9} + \varepsilon^{-3})^{124}$
3	37	1080	$x^4 + 1$	$x^5 + x^4 + x^2 + x + 1$	$(\varepsilon^{14} + \varepsilon^{-19} + \varepsilon^{155})^{39}$
4	129	1232	$x^4 + x^3 + x^2 + x + 1$	$x^5 + 1$	$\left(\frac{\varepsilon^{51} + \varepsilon^{-43}}{\varepsilon^{16}}\right)^{-53}$
5	23	2640	$x^4 + x^2 + 1$	$x^5 + x^4 + x^3 + x^2 + 1$	$(\varepsilon^3 + \varepsilon^{19} + \varepsilon^{-1} + \varepsilon^4)^{1246}$
6	73	2016	$x^5 + x^3 + 1$	$x^4 + x^3 + 1$	$(\varepsilon^4(\varepsilon^5 + \varepsilon^7))^{14}$
7	43	1740	$x^4 + x + 1$	$x^5 + x^4 + x^3 + x + 1$	$(\varepsilon^{-11}(\varepsilon^{-51} + \varepsilon^{-65}))^{32}$
8	137	660	$x^5 + x^4 + x^3 + x + 1$	$x^3 + x^2 + 1$	$(\varepsilon^{15} + \varepsilon^{-25} + \varepsilon^{436})^{-34}$
9	29	648	$x^5 + x^4 + x^2 + x + 1$	$x^4 + x^2 + 1$	$(\varepsilon^{65} + \varepsilon^{-75} + \varepsilon^{15})^{30}$
10	19	1280	$x^5 + 1$	$x^4 + x^3 + x^2 + x + 1$	$(\varepsilon^{1286} + \varepsilon^{1895} + \varepsilon^{1654})^{2658}$

1.4. Эллиптические кривые

Эффективные средства защиты, обеспечивающие аутентификацию сторон и целостность информации, играют важную роль в системах передачи данных. Так, например, алгоритм цифровой подписи ECDSA, используемый, в частности, в современных системах блокчейн, основан на криптографии эллиптических кривых (ECC – Elliptic-Curve Cryptography) [14].

1.4.1. Принципы построения эллиптических кривых

Эллиптическая кривая, или кривая Вейерштрасса представляет собой множество точек, связанных уравнением [15]:

$$y^2 = x^3 + ax + b, \quad (14)$$

где коэффициенты a и b являются вещественными и удовлетворяют следующему условию:

$$4a^3 + 27b^2 \neq 0. \quad (15)$$

Для примера построим эллиптические кривые на основе (14) с параметрами, удовлетворяющими условию (15): $\{a = -1; b = 0\}$ и $\{a = 0; b = 0\}$ (рис. 4 и 5).

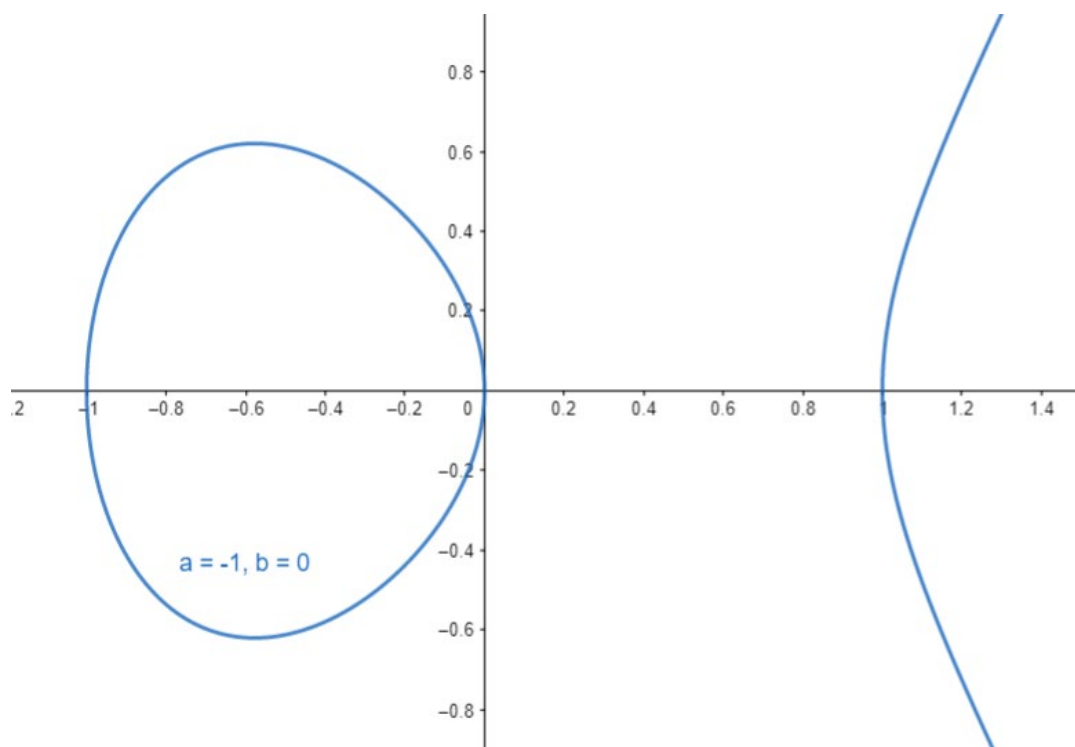


Рис. 4. Эллиптическая кривая $y^2 = x^3 - x$

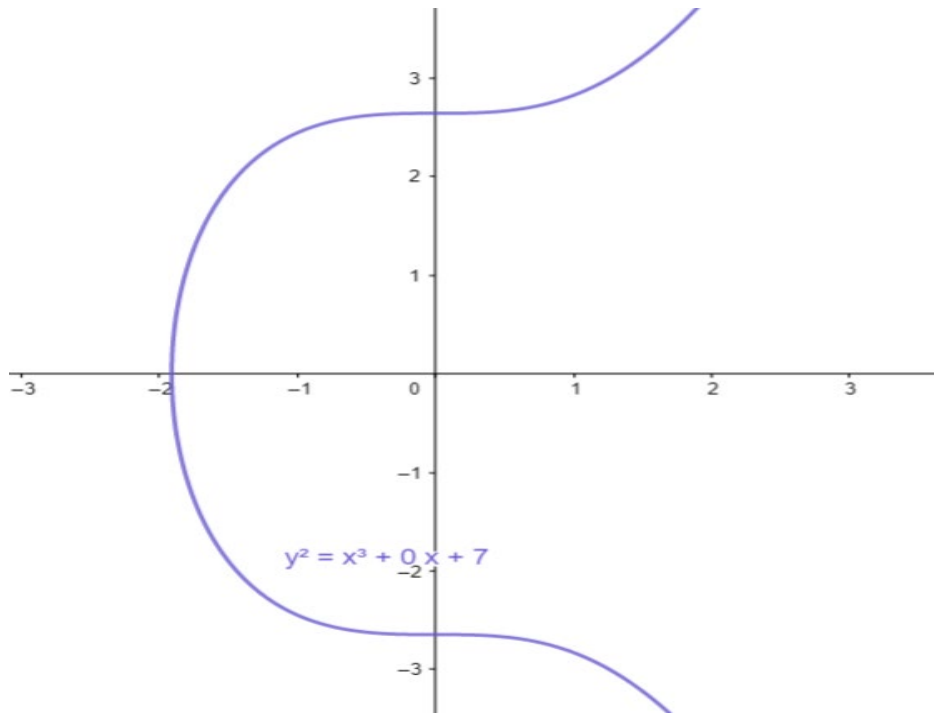


Рис. 5. Эллиптическая кривая $y^2 = x^3 + 7$

Также для примера построим кривые, для которых не выполняется условие (15): $\{a = 0; b = 0\}$ и $\{a = -3; b = 2\}$ (рис. 6 и 7).

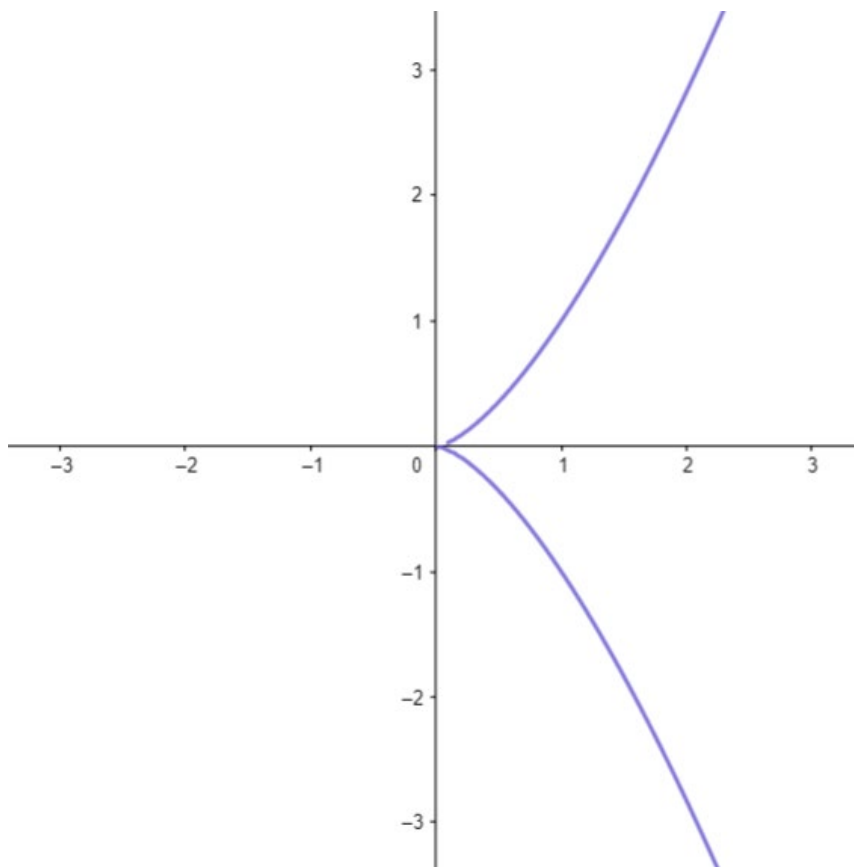


Рис. 6. Кривая $y^2 = x^3$ с точкой возврата

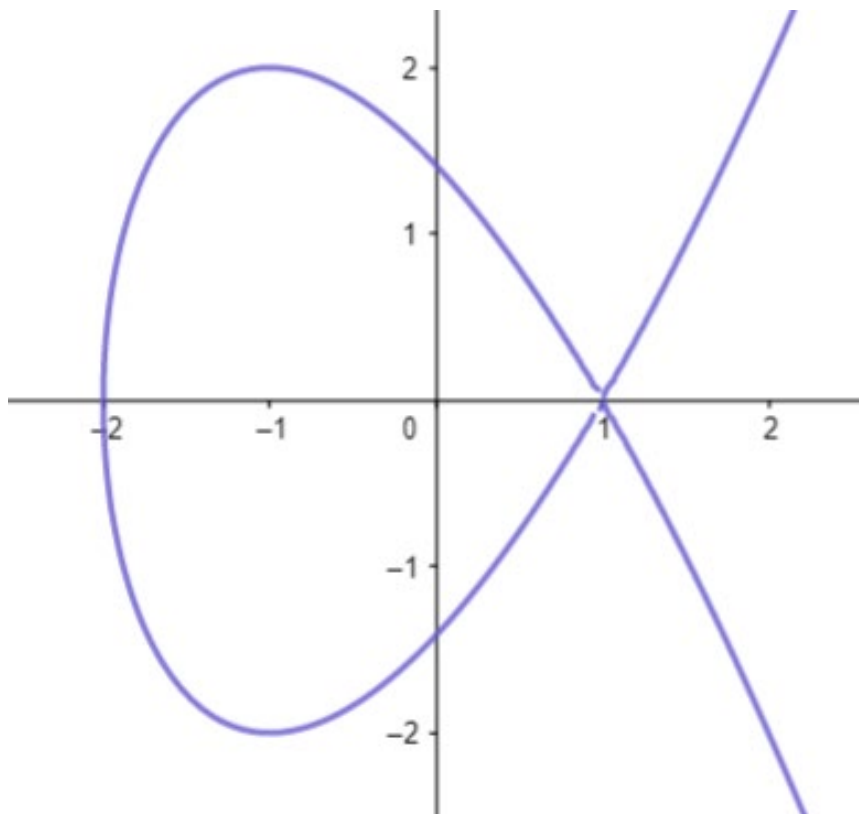


Рис. 7. Кривая $y^2 = x^3 - 3x + 2$ с самопересечением

Из графиков (рис. 6 и 7) видно, что данные кривые содержат некоторые особые точки (возврата и пересечения). Так как эти кривые не удовлетворяют (15), то не принадлежат классу кривых Вейерштрасса.

1.4.2. Точки эллиптической кривой

Лежащие на эллиптической кривой точки составляют аддитивную абелеву группу. Нулевой элемент, выполняющий роль аддитивной единицы, будет соответствовать здесь некоторой бесконечно удаленной точке на эллиптической кривой. Эту точку принято называть «точкой в бесконечности». Таким образом, обратным элементом для каждой точки данной кривой будет другая точка этой же кривой, расположенная симметрично относительно оси абсцисс [16].

Так как множество точек эллиптической кривой составляют аддитивную группу, то к ним применима операция сложения. При этом каждая точка, расположенная на плоскости, задается двумя координатами.

Пусть известны две точки эллиптической кривой: $P = (x_1, y_1)$ и $Q = (x_2, y_2)$. Требуется найти некоторую третью точку $R = (x_3, y_3)$, для которой будет справедливо равенство:

$$P + Q = R. \quad (16)$$

Геометрически операция (16) может иметь вид, представленный на рис. 8.

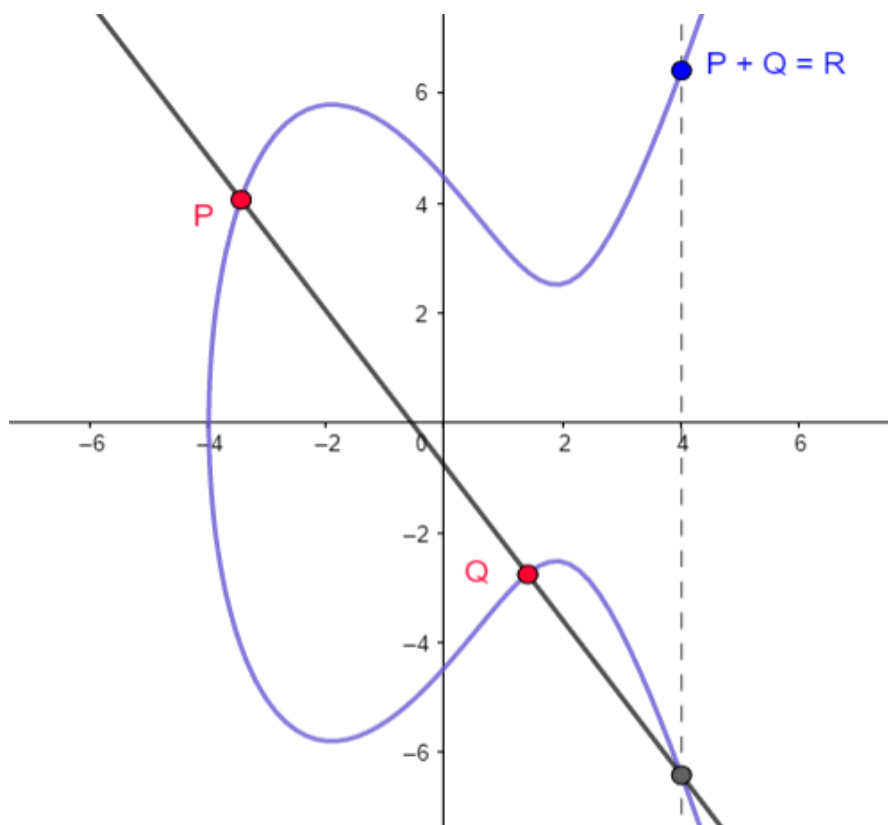


Рис. 8. Сложение двух точек эллиптической кривой

Подобный результат сложения (рис. 8) объясняется тем, что сумма трех точек, лежащих на одной прямой, равна нулю: $P + Q + (-R) = 0$.

Значения самих координат точки R определяются системой:

$$\begin{cases} x_3 = s^2 - x_1 - x_2; \\ y_3 = s(x_1 - x_3) - y_1, \end{cases} \quad (17)$$

где s – наклон прямой, которая проходит через точки эллиптической кривой P и Q (рис. 8).

Величина этого наклона вычисляется из равенства:

$$s = \frac{y_1 - y_2}{x_1 - x_2}. \quad (18)$$

Используя формулы (17) и (18), вычислим координаты точки $R = (x_3, y_3)$, которая принадлежит некоторой эллиптической кривой, например $y^2 = x^3 - 7x + 10$, и является суммой двух произвольных точек этой же кривой с известными координатами: $P = (x_1, y_1) = (1, 2)$ и $Q = (x_2, y_2) = (3, 4)$.

Значение наклона определяется равенством (18):

$$s = \frac{y_1 - y_2}{x_1 - x_2} = \frac{2 - 4}{1 - 3} = 1.$$

Подставляя значение s в систему уравнений (17), вычислим координаты точки R :

$$\begin{aligned} x_3 &= 1^2 - 1 - 3 = -3; \\ y_3 &= 1(1 + 3) - 2 = 2. \end{aligned}$$

Результат сложения точек P и Q можно представить графически (рис. 9).

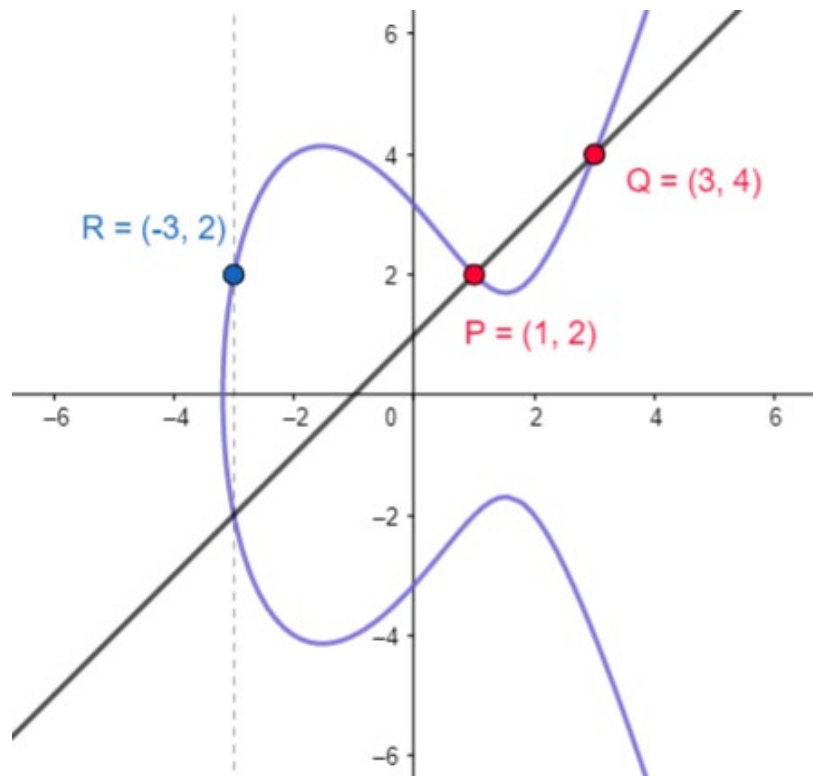


Рис. 9. Сложение точек эллиптической кривой $P = (1, 2)$ и $Q = (3, 4)$

Не составляет труда убедиться в том, что точка R принадлежит данной эллиптической кривой. Для этого подставим координаты x_3 и y_3 в уравнение этой кривой и сравним правую и левую части:

$$\begin{aligned} y_3^2 &= x_3^3 - 7x_3 + 10; \\ 2^2 &= (-3)^3 - 7 \cdot (-3) + 10; \\ 4 &= 4. \end{aligned}$$

Очевидным можно считать то, что результатом сложения двух симметрично расположенных относительно оси абсцисс точек эллиптической кривой будет аддитивная единица, то есть нулевой элемент абелевой группы. То есть для $P = (x_1, y_1)$ и $Q = (x_1, -y_1)$ получим $R = P + Q = P + (-P) = 0$.

1.4.3. Удвоение точки эллиптической кривой

При сложении двух точек эллиптической кривой возможен вариант, когда точка складывается сама с собой, то есть выполняется операция ее удвоения:

$$R = P + P = 2P. \quad (19)$$

Геометрически операция (19) имеет вид, представленный на рис. 10.

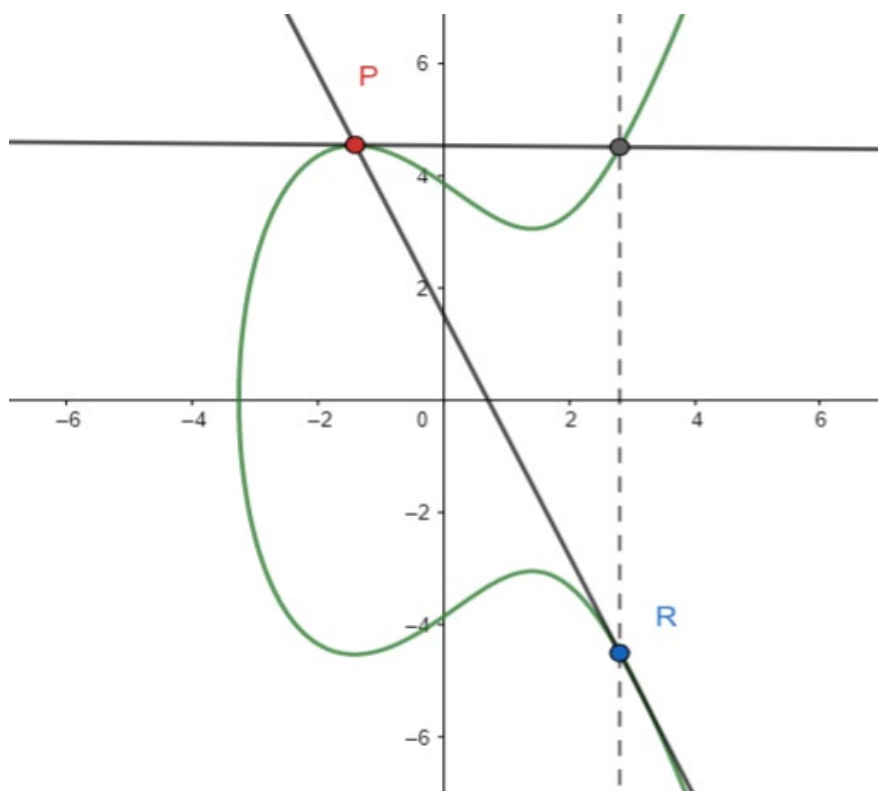


Рис. 10. Удвоение точки эллиптической кривой

Согласно графику (рис. 10), искомая точка R лежит на касательной к эллиптической кривой. При этом точка R является отраженной относительно оси абсцисс для точки пересечения данной касательной с построенной кривой.

Координаты точки R могут быть вычислены при помощи той же системы (17), в то время как для определения наклона s необходимо будет использовать несколько другое выражение:

$$s = \frac{3x_1^2 + a}{2y_1}, \quad (20)$$

где a – параметр из уравнения эллиптической кривой.

В любом случае система, определяющая значение координат удвоенной точки эллиптической кривой, принимает следующий вид:

$$\begin{cases} x_3 = s^2 - 2x_1; \\ y_3 = s(x_1 - x_3) - y_1. \end{cases} \quad (21)$$

Произведем вычисление удвоенной точки $P(1, 2)$, которая принадлежит эллиптической кривой $y^2 = x^3 - 7x + 10$.

Определим наклон при помощи (20):

$$s = \frac{3 \cdot 1^2 - 7}{2 \cdot 2} = -\frac{4}{4} = -1.$$

Подставим полученный результат s в систему (21):

$$\begin{cases} x_3 = (-1)^2 - 2 \cdot 1 = -1; \\ y_3 = (-1) \cdot (1 + 1) - 2 = -4. \end{cases}$$

Таким образом, была определена удвоенная точка $R = (-1, -4)$.

1.4.4. Скалярное умножение точки эллиптической кривой

Теперь введем понятие скалярного умножения точки, которая принадлежит эллиптической кривой:

$$R = nP = \sum_{i=1}^n P, \quad (22)$$

где n – натуральное число.

На практике операция (22) может быть эффективно реализована при помощи алгоритма удвоения-сложения [17]. В данном случае число n может быть представлено в виде суммы степеней двойки, т. е. фактически переводится в двоичную систему счисления. Приведем пример для $n = 99$.

$$n = 99_{10} = 1100011_2 = 2^6 + 2^5 + 2^1 + 2^0.$$

Теперь операцию (22) для $n = 99$ можно будет представить в виде:

$$nP = 99P = 2^6P + 2^5P + 2^1P + 2^0P = 2^6P + 2^5P + 2P + P.$$

Таким образом, для случая $n = 99$ данный алгоритм реализуется следующим образом:

P известно	//слагаемое $S_1=P$
1. Удвоение $P \Rightarrow$ Вычисление $2P$	//слагаемое $S_2=2P$
2. Вычисление суммы $S=S_1+S_2$	// $P+2P$
3. Удвоение $2P \Rightarrow$ Вычисление 2^2P	
4. Удвоение $2^2P \Rightarrow$ Вычисление 2^3P	
5. Удвоение $2^3P \Rightarrow$ Вычисление 2^4P	
6. Удвоение $2^4P \Rightarrow$ Вычисление 2^5P	//слагаемое $S_3=2^5P$
7. Вычисление суммы $S=S+S_3$	// $P+2P+2^5P$
8. Удвоение $2^5P \Rightarrow$ Вычисление 2^6P	//слагаемое $S_4=2^6P$
9. Вычисление суммы $S=S+S_4$	// $P+2P+2^5P+2^6P=99P$

В результате вместо $n - 1 = 98$ операций сложения в рамках данного алгоритма было произведено 6 удвоений и 3 сложения.

1.4.5. Эллиптические кривые над конечным полем

Стоит подчеркнуть, что в поле над элементами выполняются две операции: сложение и умножение, а также присутствуют аддитивная и мультипликативная единицы, соответственно нулевой и единичный элементы, которые обеспечивают обратимость остальных элементов поля по сложению и умножению [18].

Результат операций сложения и умножения в конечном поле $GF(p)$ приводится по $\text{mod } p$. Это же правило действует для точек эллиптической кривой над конечным полем, уравнение которой приобретает вид [16]:

$$y^2 = x^3 + ax + b \pmod{p}, \quad (23)$$

где коэффициенты a и b удовлетворяют условию:

$$4a^3 + 27b^2 \not\equiv 0 \pmod{p}. \quad (24)$$

Результат сложения двух точек $P = (x_1, y_1)$ и $Q = (x_2, y_2)$ такой эллиптической кривой также будет приводиться по $\text{mod } p$:

$$\begin{cases} x_3 = s^2 - x_1 - x_2 \pmod{p}; \\ y_3 = s(x_1 - x_3) - y_1 \pmod{p}. \end{cases} \quad (25)$$

Значение s в (25) вычисляется в соответствии с правилом:

$$\begin{cases} s = \frac{y_1 - y_2}{x_1 - x_2} \pmod{p}, & P \neq Q; \\ s = \frac{3x_1^2 + a}{2y_1} \pmod{p}, & P = Q. \end{cases} \quad (26)$$

Приведем пример со сложением двух точек, принадлежащих эллиптической кривой $y^2 = x^3 - 7x + 10 \pmod{29}$. Сначала убедимся в том, что данная кривая может существовать над полем $\text{GF}(29)$. Для этого проверим выполнение условия (24) при $a = -7$ и $b = 10$:

$$4 \cdot (-7)^3 + 27 \cdot (10)^2 = 1328 \not\equiv 0 \pmod{29}.$$

Произведем сложение точек $P = (1, 2)$ и $Q = (5, 10)$.

Следуя правилу (26), вычислим параметр s :

$$s = \frac{2 - 10}{1 - 5} \pmod{29} = \frac{-8}{-4} \pmod{29} = 2.$$

Используя (25), определим координаты точки $R = P + Q$:

$$x_3 = 2^2 - 1 - 5 = -2 = 27 \pmod{29};$$

$$y_3 = 2 \cdot (1 - 27) - 2 = -54 = 4 \pmod{29}.$$

Таким образом, получим точку $R = (27, 4)$. Убедимся, что данная точка принадлежит данной эллиптической кривой, подставив значения ее координат в уравнение (23):

$$y_3^2 = x_3^3 - 7x_3 + 10 \pmod{29};$$

$$4^2 = 27^3 - 7 \cdot 27 + 10 \pmod{29};$$

$$16 = 19504 \pmod{29};$$

$$16 = 16.$$

1.4.6. Порядок группы точек эллиптической кривой

Точки эллиптической кривой над конечным полем $\text{GF}(p)$ составляют циклически замкнутую группу, имеющую порядок N , который в соответствии с теоремой Хассе удовлетворяет неравенству [19]:

$$|N - (p + 1)| \leq 2\sqrt{p}. \quad (27)$$

Выражение (27) можно записать в виде двойного неравенства:

$$p+1-2\sqrt{p} \leq N \leq p+1+2\sqrt{p}, \quad (28)$$

которое говорит о том, что число точек эллиптической кривой над конечным полем лежит в некотором диапазоне.

Рассмотрим построенную ранее эллиптическую кривую $y^2 = x^3 - 7x + 10$ над полем $\text{GF}(29)$. Используя выражение (28), не составляет труда определить, что число ее точек лежит в диапазоне от 19 до 40:

$$p+1-2\sqrt{p} = 29+1-2\sqrt{29} \approx 19;$$

$$p+1+2\sqrt{p} = 29+1+2\sqrt{29} \approx 40.$$

Стоит отметить, что практическое применение ЕСС в криптосистемах, обеспечивающих цифровую подпись, предполагает использование эллиптических кривых с заранее известным значением N , которые присутствуют в списке Национального института стандартов и технологий США (NIST – The National Institute of Standards and Technology) [20].

1.4.7. Циклические подгруппы точек эллиптической кривой

Известно, что точки эллиптических кривых над полем $\text{GF}(p)$ образуют циклическую подгруппу вида $\{0, P, 2P, 3P, \dots\}$, где P выступает в роли некоторой базовой или порождающей точки, которая формирует остальные точки данной подгруппы путем умножения на целочисленный коэффициент $n = 0, 1, 2, \dots$, для которого не определена верхняя граница.

Рассмотрим пример с эллиптической кривой $y^2 = x^3 + 7 \pmod{29}$ над полем $\text{GF}(29)$. Выберем произвольную базовую точку $P = (6, 7)$. Построим подгруппу, используя вышеописанную методику:

$$n = 0: 0P = 0;$$

$$n = 1: 1P = (6, 7);$$

$$n = 2: 2P = (12, 13);$$

$$n = 3: 3P = (12, 16);$$

$$n = 4: 4P = (6, 22);$$

$$n = 5: 5P = 0;$$

$$n = 6: 6P = P = (6, 7);$$

...

Данный пример наглядно демонстрирует процедуру построения циклически замкнутой подгруппы, которая в данном случае будет состоять только из пяти точек: $\{0, P, 2P, 3P, 4P\}$. При этом нетрудно убедиться, что такое

множество является аддитивной абелевой группой, для которой будут справедливы соответствующие аксиомы.

В соответствии с теоремой Лагранжа порядок подгруппы является делителем порядка группы. Так, в примере с кривой $y^2 = x^3 + 7 \pmod{29}$, которая была обозначена выше, порядок группы точек равен 30. Порождающая точка $P = (6, 7)$, в свою очередь, позволяет построить подгруппу из пяти элементов. Число 5 действительно является делителем числа 30, что соответствует теореме Лагранжа.

Таким образом, алгоритм определения порядка подгруппы с базовой точкой P , которая входит в состав группы точек той или иной эллиптической кривой, сводится к последовательности следующих действий:

1) вычисление порядка всей группы точек эллиптической кривой над полем $\text{GF}(p)$;

2) нахождение делителей порядка группы;

3) вычисление произведений базовой точки P и делителей в порядке их увеличения. Результат одного из произведений, равный значению 0, говорит о соответствии текущего делителя порядку подгруппы точек, построенной на основе базовой точки.

Приведем пример с произвольной базовой точкой $P = (24, 6)$ на эллиптической кривой $y^2 = x^3 + 4x + 7 \pmod{29}$. Порядок группы точек самой кривой с учетом «точки в бесконечности» составляет 32 элемента. При этом число 32 имеет следующие делители: 1, 2, 4, 8, 16, 32. Пользуясь вышеописанным подходом, будем последовательно (или даже параллельно) умножать точку P на коэффициенты, равные делителям, пока не получим нулевой результат произведения:

$$1P = (24, 6) \neq 0;$$

$$2P = (4, 0) \neq 0;$$

$$4P = 0..$$

Теперь можно сделать вывод, что базовая точка $P = (24, 6)$ порождает циклически замкнутую подгруппу, состоящую из четырех точек. И не составляет труда убедиться:

$$n = 0: 0P = 0;$$

$$n = 1: 1P = (24, 6);$$

$$n = 2: 2P = (4, 0);$$

$$n = 3: 3P = (24, 23);$$

$$n = 4: 4P = 0;$$

$$n = 5: 5P = P = (24, 6);$$

...

Необходимо отметить роль базовой точки подгруппы в реализации схем шифрования с открытым ключом и цифровой подписи, основанных на ECC. Сам по себе классический подход к выбору базовой точки, которая потом ляжет в основу открытого ключа, следующий:

- 1) определить N – порядок группы точек эллиптической кривой над полем $GF(p)$;
- 2) выбрать n – делитель N и при этом большое простое число;
- 3) вычислить кофактор подгруппы точек эллиптической кривой $h = N/n$;
- 4) выбрать случайную точку P на этой эллиптической кривой;
- 5) вычислить $G = hP$;
- 6) проверить условие. Если $G = 0$, то вернуться к шагу 4, иначе G подходит на роль базовой точки подгруппы точек эллиптической кривой над полем $GF(p)$. При этом порядок такой подгруппы будет равен значению n .

Смысл приведенного выше алгоритма строится на том, что для каждой точки эллиптической кривой будет справедливо равенство:

$$NP = 0. \quad (29)$$

Выражение (29) логично, так как порядок группы точек N делит себя сам. Вместе с тем выражение, определяющее кофактор подгруппы $h = N/n$, можно преобразовать в $N = nh$ и подставить в (29):

$$nhP = 0. \quad (30)$$

Учитывая равенство $G = hP$, выражение (1.30) примет вид $nG = 0$, что, в свою очередь, позволит считать G базовой точкой, которая способна построить подгруппу, состоящую из n точек эллиптической кривой.

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

- 1) выбрать из табл. 3 в соответствии с номером своего варианта исходные данные;
- 2) записать уравнение эллиптической кривой с параметрами a и b . Проверить, является ли полученная эллиптическая кривая особой;
- 3) проверить, принадлежат ли точки P_1 и Q_1 данной эллиптической кривой;
- 4) вычислить результат сложения точек P_1 и Q_1 . Проверить, принадлежит ли полученная точка данной эллиптической кривой;

5) вычислить результат удвоения точки P_1 . Проверить, принадлежит ли полученная точка данной эллиптической кривой;

6) вычислить результат скалярного умножения точки P_1 на n по алгоритму удвоения-сложения. Проверить, принадлежит ли полученная точка данной эллиптической кривой;

7) записать уравнение эллиптической кривой над конечным полем $GF(p)$, используя параметры a, b, p из табл. 3. Проверить, является ли полученная эллиптическая кривая особой;

8) проверить, принадлежат ли точки P_2 и Q_2 данной эллиптической кривой;

9) вычислить результат сложения точек P_2 и Q_2 . Проверить, принадлежит ли полученная точка данной эллиптической кривой;

10) вычислить результат удвоения точки P_2 . Проверить, принадлежит ли полученная точка данной эллиптической кривой;

11) вычислить результат скалярного умножения точки P_2 на n по алгоритму удвоения-сложения. Проверить, принадлежит ли полученная точка данной эллиптической кривой.

Таблица 3

Варианты заданий согласно номеру в журнале обучающихся

№ вар.	a	b	P_1	P_2	Q_1	Q_2	n	p
0	0	7	(3.07232, 6)	(5, 36)	(2.62074, 5)	(57, 30)	175	97
1	0	1	(0, 1)	(23, 14)	(2, 3)	(47, 23)	200	73
2	13	-5	(1, 3)	(19, 7)	(6, 17)	(8, 3)	153	43
3	-3	1	(3, 4.3589)	(12, 10)	(5, 10.53565)	(11, 1)	97	59
4	5	-5	(14, 53)	(8, 11)	(1, 1)	(21, 29)	231	71
5	-25	30	(-5, -5.47723)	(43, 4)	(1.23542, 1)	(16, 35)	53	61
6	-2	1	(2.5, -3.40955)	(33, 4)	(-1, 1.41421)	(18, 30)	127	83
7	11	20	(2.5, 7.94512)	(20, 9)	(0.44646, 5)	(10, 33)	225	41
8	-5	10	(-2, 3.4641)	(17, 11)	(2.6891, -4)	(73, 34)	101	89
9	-13	0	(-3, -3.4641)	(19, 10)	(4.5, 5.71183)	(31, 26)	201	37

2. ЦЕЛОСТНОСТЬ ДАННЫХ

Популярная сегодня технология блокчейн является идеальным примером демонстрации принципа поддержания целостности информации. Основным требованием здесь является доверие всех узлов сети блокчейн к тому, что содержащаяся в нем информация не подверглась искажению.

Вторым важнейшим моментом при организации блокчейн является подтверждение узлами своего авторства на входящие в состав блокчейн блоки информации, в подлинности которых должны быть также уверены и остальные узлы сети.

2.1. Функции хеширования

Хеш-функция (от англ. *hash function* от *hash* – «превращать в фарш») – математическая функция, преобразующая массив входных данных произвольного размера в выходную строку определенного размера в соответствии с некоторым алгоритмом.

Преобразование, выполняемое хеш-функцией, называется *хешированием*. Исходные (входные) данные называются входным массивом. Результат преобразования (выходные данные) называется *хешем*.

Таким образом, цель функции хеширования – представить данные произвольного размера в краткой форме фиксированной длины. При этом функции хеширования предназначены для защиты целостности данных.

Основным механизмом реализации функций хеширования являются итерированные техники построения, в основе которых, как правило, лежат схемы с обратными связями.

Имеет смысл привести требования, предъявляемые к современным функциям хеширования, которые должны:

- 1) обрабатывать любой текст и сжимать его до сообщения заданного размера (как правило, 128–512 бит);
- 2) быть *односторонними*, т. е. не позволять однозначно восстановить исходное сообщение за заданное время;
- 3) быть *быстрыми*, т. е. не оказывать существенного влияния на передачу и обработку самих сообщений, какими бы длинными они ни были.

Определим также свойства функций хеширования, которые в идеале не должны противоречить перечисленным выше требованиям [14]:

- вычислительная неразрешимость;
- вторичная вычислительная неразрешимость;
- устойчивость к коллизиям.

Пусть, x_1 и x_2 – некоторые сообщения, регистрируемые в информационной системе, h – функция хеширования, y_1 и y_2 – хеши, связанные с сообщениями через функцию хеширования следующим образом:

$$y_1 = h(x_1); y_2 = h(x_2).$$

Тогда свойства функции h можно представить графически (рис. 11).

Вычислительная неразрешимость тождественна односторонности, т. е. подчеркивает низкую вероятность возможности вычисления нахождения x из y . Это свойство считается сильной стороной любой функции хеширования.

Вторичная вычислительная неразрешимость, в свою очередь, считается слабой стороной функции хеширования. В данном случае речь идет о высокой вероятности существования такого информационного сообщения, которое приведет к идентичному хешу. То есть разные значения x_1 и x_2 способны привести к одному хешу. Данное свойство говорит о слабой устойчивости функции хеширования к коллизиям.

И наконец, непосредственно устойчивость к коллизиям, сводящая к минимуму вероятность возникновения предыдущей ситуации. Такое свойство является сильной стороной функции хеширования.

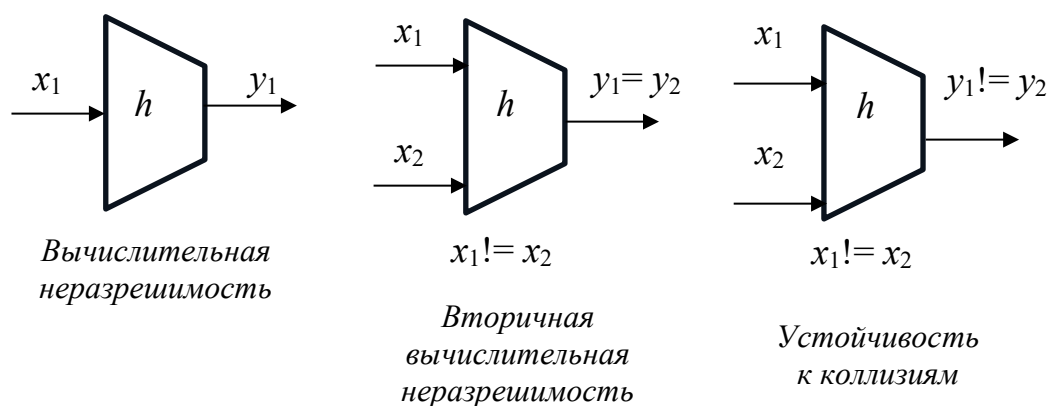


Рис. 11. Свойства функций хеширования

Конечно, любую хеш-функцию нельзя считать идеальной, то есть полностью односторонней и максимально устойчивой к коллизиям, так как размер исходного сообщения, как правило, всегда значительно больше по объему, чем его хеш. Тем не менее любая *хорошая* хеш-функция обладает так называемым *лавинным эффектом*. Это означает, что любое малейшее изменение входного текста неизбежно приводит к другому хешу.

Функции хеширования можно разделить на следующие группы [14]:

- MD (дайджест сообщения – Message Digest);
- алгоритмы безопасного хеширования;

- RIPEMD;
- Whirlpool.

Протоколы типа MD (MD4 и MD5) в настоящее время считаются устаревшими и небезопасными. Что касается алгоритмов так называемого безопасного хеширования (SHA – Secure Hash Algorithms), то они, в свою очередь, делятся на три категории:

- SHA0 (устаревший);
- SHA1 (устаревший);
- SHA2 (группа функций: SHA-224, SHA-256, SHA-384, SHA-512);
- SHA3 (группа функций: SHA3-224, SHA3-256, SHA3-384).

В частности, сеть блокчейн криптовалюты Bitcoin использует алгоритмы хеширования SHA-256 и RIPEMD160, а в Ethereum применяется SHA3.

2.2. Хеширование в OpenSSL

Общедоступная программная библиотека OpenSSL позволяет организовать эффективную работу с различными хеш-функциями. Список доступных алгоритмов хеширования выводится при помощи команды:

```
$ openssl dgst -list
```

Supported digests:

-blake2b512	-blake2s256	-md4
-md5	-md5-sha1	-ripemd
-ripemd160	-rmd160	-sha1
-sha224	-sha256	-sha3-224
-sha3-256	-sha3-384	-sha3-512
-sha384	-sha512	-sha512-224
-sha512-256	-shake128	-shake256
-sm3	-ssl3-md5	-ssl3-sha1

Процедура хеширования запускается командой **openssl dgst** с соответствующими параметрами:

```
openssl dgst [алгоритм] [путь к файлу]
```

Данная команда должна отобразить на экране значение хеша файла в шестнадцатеричном формате. Для примера создадим некоторый текстовый файл *test.txt* и запишем в него строку «blockchain». Вычислим хеш данного файла, используя алгоритм SHA3-256:

```
$ cat test.txt
```

```
Blockchain
```

```
$ openssl dgst -sha3-256 test.txt
```

```
SHA3-256(test.txt)=  
45740502697d57cbc7e6522372d3247adf1ab8f1cdb0cda1f20a022bf3e153  
d0
```

В данном случае символы строки «blockchain» в файле *test.txt* сначала переводятся в двоичный вид в соответствии с кодировкой ASCII, а затем для них вычисляется хеш-функция.

Используя утилиту **echo**, на вход **openssl dgst** можно также передать и отдельную строку при помощи команды с соответствующими параметрами:

```
$ echo -n [входная строка] | openssl dgst [алгоритм]
```

В данном случае ключ **-n** говорит о том, что к целевой строке не будет добавлен символ конца строки «\n». Таким образом, хеш строки «blockchain» по алгоритму SHA3-256 будет следующим:

```
$ echo -n blockchain | openssl dgst -sha3-256  
  
(stdin)=  
45740502697d57cbc7e6522372d3247adf1ab8f1cdb0cda1f20a022bf3e153d0
```

Как и ожидалось, полученный хеш строки и вычисленный ранее хеш текстового файла совпадают. При этом в процессе вычисления хеша для файла необходимо учитывать особенности различных текстовых редакторов, которые могут автоматически добавлять символ конца строки, что в итоге приведет к изменению хеша.

Для вывода результата хеширования в бинарном формате к команде **openssl dgst** необходимо добавить параметр **-binary**:

```
$ echo -n "blockchain" | openssl dgst -sha3-256 -binary
```

```
Eti}WR#r$zF~+S
```

Ключ **-binary** обеспечивает получение хеша в бинарном виде, иногда это необходимо для его вторичного хеширования. Таким образом, двойное хеширование строки «blockchain» алгоритмом SHA3-256 будет выглядеть следующим образом:

```
$ echo -n "blockchain" | openssl dgst -sha3-256 -binary |  
openssl dgst -sha3-256  
  
(stdin)=  
4cdc2bfc3db1d62ef1e09b6db5b527909bc485684a92ff079d2ef17849b82b29
```

Что касается сети блокчейн или мессенджеров, то обычно здесь сначала производится конкатенация значений всех полей в общую строку, для которой затем вычисляется хеш. Например, пусть имеется некоторая транзакция мессенджера:

```
To: 0x0000006daea1723962647b7e189d311d757fb793
From: 0x00006196242a1d328fe4b636995e796cb6c7a2ac
Message: Hello there!
```

Поля транзакции содержат следующую информацию: *To* – адрес отправителя, *From* – адрес получателя, *Message* – передаваемое сообщение.

Символ **0x** обозначает то, что строка представлена в шестнадцатеричном формате. В процессе конкатенации данный символ должен быть изъят. Поле *Message* содержит данные в текстовом формате. Перед конкатенацией они также должны быть переведены в шестнадцатеричный формат:

48656c6c6f207468657265210a.

В результате последовательной конкатенации данных, содержащихся в полях транзакции, будет получена следующая строка:

0000006daea1723962647b7e189d311d757fb79300006196242a1d328fe4b636995e796cb6c7a2ac48656c6c6f207468657265210a.

Процесс хеширования с конкатенацией требует обязательного соблюдения последовательности конкатенируемых строк. Изменение порядка хешируемых строк неизбежно приведет на выходе к другому хешу.

Заметим, что хешируемые данные должны быть представлены не в текстовой, а в шестнадцатеричной форме. Поэтому требуется перевести шестнадцатеричную строку в бинарный вид при помощи соответствующей программы, например утилиты **xxd** с параметрами **-r** и **-p**. По умолчанию данная утилита предназначена для перевода бинарных данных в шестнадцатеричный вид. Однако параметр **-r** запускает обратную операцию. Параметр **-p**, в свою очередь, обеспечивает вывод шестнадцатеричных данных в виде текста.

Вычислим хеш полученной ранее шестнадцатеричной строки по произвольному алгоритму хеширования, например MD5, совместив команды **xxd -r -p** и **openssl dgst**:

```
$ echo -n
0000006daea1723962647b7e189d311d757fb79300006196242a1d328fe4b6
36995e796cb6c7a2ac48656c6c6f207468657265210a | xxd -r -p |
openssl dgst -md5
```

```
(stdin)= 483b8f983ac1854a44df8d3532b4e637
```

Подобная конкатенация строк требует обязательного преобразования их численных значений в шестнадцатеричный формат. Преобразовать десятичное число в шестнадцатеричный формат, например, можно при помощи команды **printf**:

```
$ printf "%x" [десятичное число]
```

Преобразование текстовой строки в шестнадцатеричный формат, в свою очередь, реализуется командами **xxd -p** и **tr -d "\n"**:

```
$ echo -n [текстовая строка] | xxd -p | tr -d "\n"
```

Параметр **-p** обеспечивает вывод шестнадцатеричной строки в «чистом виде». Команда **tr -d "\n"** удаляет из шестнадцатеричной строки символ конца строки.

Рассмотрим пример с вычислением хеша для некоторой транзакции из сферы образования. Пусть эта транзакция будет включать в себя следующие данные:

- информацию о студенте (*Student*: Ф. И. О.);
- информацию об учебном заведении (*Academy*: название вуза);
- пройденное академическое время (*Hours*: количество часов);
- идентификатор дипломной работы (*Diploma ID*: уникальный идентификатор);
- публичный ключ для проверки подписи идентификатора дипломной работы (*Public Key*: ключ);
- научный руководитель дипломной работы (*Supervisor*: Ф. И. О.);
- состояние дипломной работы (*Status*: статус).

Заполним поля транзакции произвольными данными:

```
Student: Бред Питт
Academy: СПбГУТ им. М.А. Бонч-Бруевича
Hours: 3
Diploma ID: 0x626c6f636b636861696e0a
Public key: 0x7075626c6963206b65790a
Supervisor: Уилл Смит
Status: Защищена
```

Произведем последовательное преобразование полей транзакции в шестнадцатеричный формат с добавлением результатов в текстовый файл *diploma.txt*, тем самым конкатенируя строки:

```
$ echo -n Бред Питт | xxd -p | tr -d "\n" >> diploma.txt
```

```
$ echo -n СПбГУТ им. М.А. Бонч-Бруевича | xxd -p | tr -d "\n"
>> diploma.txt
```

```
$ printf "%x" 3 >> diploma.txt

$ echo -n 626c6f636b636861696e0a >> diploma.txt

$ echo -n 7075626c6963206b65790a >> diploma.txt

$ echo -n Уилл Смит | xxd -p | tr -d "\n" >> diploma.txt

$ echo -n Защищена | xxd -p | tr -d "\n" >> diploma.txt

$ cat diploma.txt

d091d180d0b5d0b420d09fd0b8d182d182d0a1d09fd0b1d093d0a3d0a220d0
b8d0bc2e20d09c2ed0902e20d091d0bed0bdd1872dd091d180d183d0b5d0b2
d0b8d187d0b03626c6f636b636861696e0a7075626c6963206b65790ad0a3d
0b8d0bbd0bb20d0a1d0bcd0b8d182d097d0b0d189d0b8d189d0b5d0bdd0b0
```

Используя произвольный алгоритм, например, RIPEMD-160, вычислим хеш общей строки, представленной в бинарном виде:

```
$ cat diploma.txt | xxd -r -p | openssl dgst -ripemd160

(stdin)= 5dc089df7b9e25e188201955cbd90042e8960fe1
```

Команда **cat** производит чтение содержимого файла *diploma.txt* и отправляет данные на вход команды **xxd -r -p**, которая, в свою очередь, переводит шестнадцатеричные данные из файла в бинарный формат и передает на вход команды **openssl dgst**.

Таким образом, хеш транзакции по алгоритму RIPEMD-160 будет иметь следующий вид:

c5dc089df7b9e25e188201955cbd90042e8960fe1.

2.3. Концепция блокчейн

Термин *блокчейн* (от англ. block – блок, chain – цепь) – буквально означает «цепочка блоков». Блок в данном случае выступает в качестве некоторого контейнера данных, где хранится пользовательская и служебная информация в блокчейн сети (рис. 12).

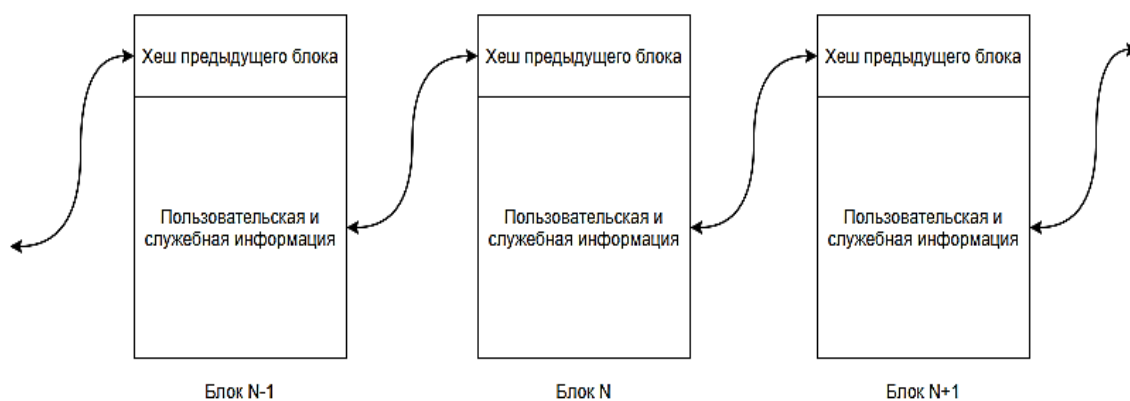


Рис. 12. Структура блокчейн

Как видно из рис. 12, технология блокчейн использует хеш-функции. Они обеспечивают связь блоков, механизмы консенсуса, проверки присутствия конкретной транзакции в блоке. Каждый блок хранит в себе хеш предыдущего блока [14]. Искривление данных в блоке неизбежно приводит к изменению его хеша, а это, в свою очередь, повлечет за собой изменение хеша предыдущего блока.

2.3.1. Транзакции в сети блокчейн

На практике служебная информация блока содержится, как правило, в заголовке блока, а пользовательская определяет транзакцию, которая как раз и защищается хешем (рис. 13).

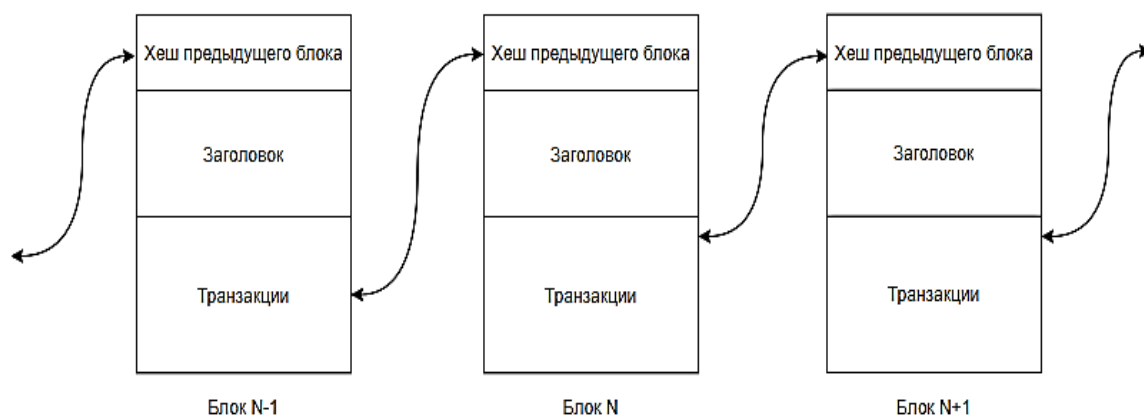


Рис. 13. Обеспечение целостности транзакций

Транзакция считается основной единицей пользовательской информации в сети блокчейн. Если речь заходит о криптовалютах, таких, например, как Bitcoin или Ethereum, то их транзакции несут в себе информацию о денежных переводах.

При создании транзакции узел сети блокчейн ставит целью доказать свою причастность к ней как владелец, используя для этого механизм

цифровой подписи. Такой подход позволяет остальным узлам сети удостовериться в подлинности транзакции.

Таким образом, каждый узел сети блокчейн имеет два ключа: публичный (открытый) и приватный (секретный). Приватный ключ тщательно хранится на стороне владельца, публичный ключ доступен всем пользователям сети блокчейн. В сети Bitcoin, например, публичный ключ определяет адрес кошелька.

Итак, безопасность транзакции опирается на два основных постулата: подтверждение авторства создавшего ее узла и сохранение целостности. Обе эти задачи обычно решаются при помощи алгоритма DSA (Digital Signature Algorithm – алгоритм проверки цифровой подписи).

Основная идея DSA состоит в том, что само информационное сообщение передается по открытому каналу связи вместе со своим хешем, который подписан цифровой подписью (рис. 14).

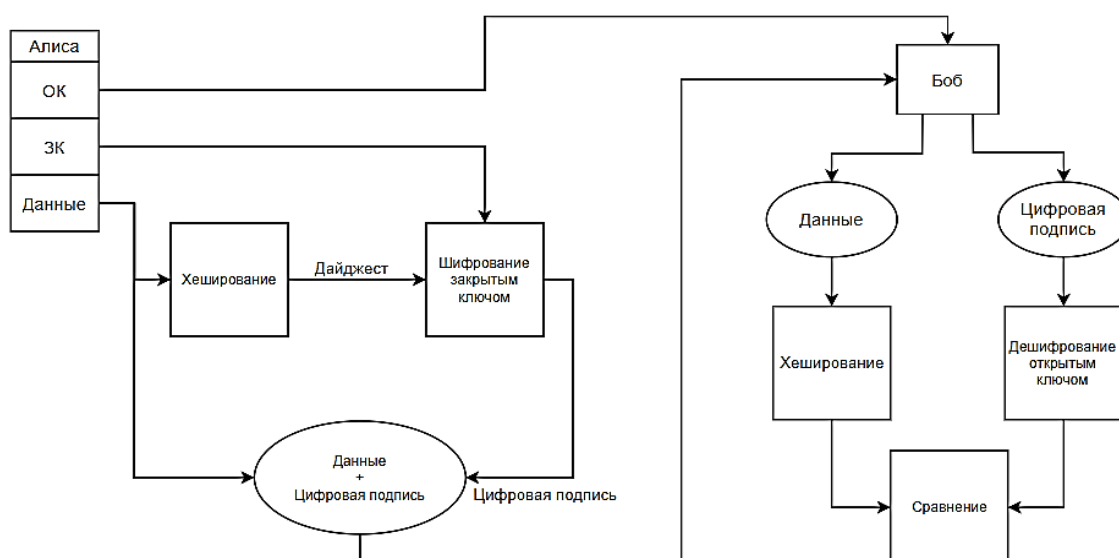


Рис. 14. Логический принцип DSA

Разберем логику данного алгоритма. Допустим, что *Алисе* необходимо подписать свою транзакцию (*Данные*). Для этого у нее имеется закрытый ключ (*ЗК*).

Также у *Алисы* имеется открытый ключ (*ОК*), который публикуется в сети и тем самым становится доступен *Бобу*.

Алиса вычисляет хеш (*Дайджест*) транзакции и производит его шифрование при помощи своего *ЗК*. Таким образом, *Алиса* получает цифровую подпись транзакции.

Далее *Алиса* производит отправку транзакции в сеть, прикрепляя к ней подписанный хеш (*Цифровая подпись*).

Боб получает транзакцию с подписанным хешем *Алисы* и должен убедиться в том, что принятым данным можно доверять: автором транзакции действительно является *Алиса* и транзакция не была искажена в процессе

передачи по сети. Для этого Боб, с одной стороны, вычисляет хеш принятой транзакции, а с другой – производит обработку цифровой подписи при помощи имеющегося у него в наличии *ОК Алисы*. Результаты этих вычислений должны быть тождественны, только в этом случае транзакция будет считаться подлинной.

Возможен также вариант, когда комплект, содержащий *Данные* и *Цифровую подпись*, перед отправкой в сеть шифруют при помощи открытого ключа получателя. Или же хеш вычисляется для *Данных*, которые уже были зашифрованы открытым ключом получателя. Однако чаще всего, когда речь идет о сети блокчейн, сами транзакции не подвергаются шифрованию и имеют общедоступную видимость.

Стоит также отметить, что современные информационные системы, в том числе многие сети блокчейн, реализуют механизм цифровой подписи на основе схемы ECDSA, где используется криптография эллиптических кривых. По сравнению с алгоритмом цифровой подписи в RSA и схемой Эль-Гамала ключи в ECDSA имеют значительно меньшую длину и обладают большей криптостойкостью.

2.3.2. Дерево Меркла

Одной из самых распространенных задач в сети блокчейн является выявление факта наличия определенной транзакции в конкретном блоке. В качестве входного параметра для запуска процедуры поиска транзакции в блоке задается ее хеш.

Последовательная проверка всех транзакций отдельно взятого блока является достаточно медленной процедурой. Для ускорения данного процесса обычно используют так называемое дерево Меркла [14].

Дерево Меркла или хешей представляет собой бинарное дерево, где каждый узел имеет только два ответвления (рис. 15).

Узлы T1–T8 нижнего слоя дерева (рис. 15) определяют хеши транзакций отдельно взятого блока. В случае нечетного числа транзакций последняя из них дублируется. Каждый последующий узел – это хеш значений двух предыдущих узлов. Так, например, узлы T1 и T2 восходят к узлу, который содержит $\text{hash}(T1+T2)$, т. е. хеш значения суммы T1 и T2.

Самый верхний узел в схеме (рис. 15) называют корнем дерева Меркла. Очевидно, что корень хранит в себе суммарный хеш значений всех узлов дерева. Применительно к данной схеме (рис. 15) хеш, содержащийся в корне дерева Меркла, можно выразить следующим образом:

$$\text{hash} \left[\begin{array}{l} \text{hash}(\text{hash}(T1+T2) + \text{hash}(T3+T4)) + \\ + \text{hash}(\text{hash}(T5+T6) + \text{hash}(T7+T8)) \end{array} \right]$$

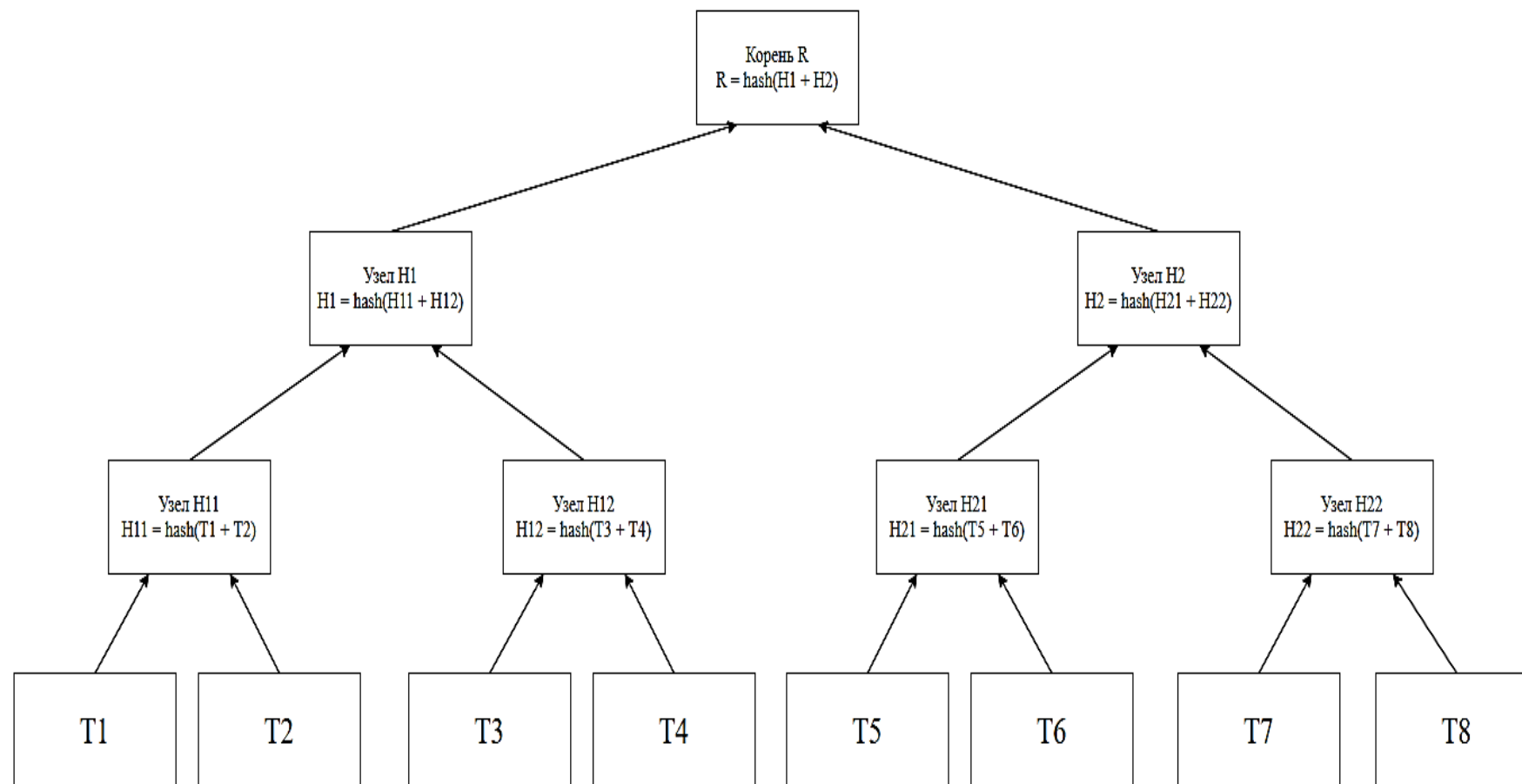


Рис. 15. Структура дерева Меркла

Рассмотрим пример построения дерева Меркла, где для определения хешей будем использовать упомянутую ранее программную библиотеку *OpenSSL*.

Пусть заданы следующие хеши транзакций некоторого блока: «a», «b» и «c». Так как число транзакций является нечетным, то произведем дублирование последнего хеша «c». В итоге нижний слой дерева Меркла будет выглядеть следующим образом (рис. 16).

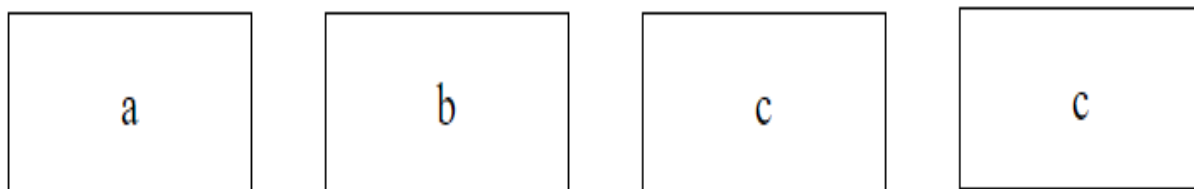


Рис. 16. Нижний слой дерева Меркла

Далее определяются хеши значений «ab» и «cc». В качестве алгоритма хеширования в данном примере предлагается использовать SHA3-256. Пусть результаты вычисления хешей для удобства будут сохраняться в файлы с названием узлов дерева.

Следует подчеркнуть, что конкатенированные значения хешей «ab» и «cc» должны быть представлены в бинарном виде. Для преобразования шестнадцатеричной строки в бинарные данные целесообразно воспользоваться утилитой **xxd** с параметрами **-r** и **-p**. Как было показано выше, утилита **xxd** преобразует бинарные данные в шестнадцатеричные. Вместе с тем, ключ **-r** обеспечивает обратное преобразование.

Вычислим хеши конкатенированных данных для узлов H1 и H2 и при помощи ключа **-out** сохраним эти значения в файлы с названиями *H1.bin* и *H2.bin* соответственно:

```
$ echo -n "ab" | xxd -r -p | openssl dgst -sha3-256 -binary -  
out H1.bin  
$ echo -n "cc" | xxd -r -p | openssl dgst -sha3-256 -binary -  
out H2.bin
```

Далее вычислим хеш корня дерева Меркла, конкатенируя данные из файлов *H1.bin* и *H2.bin* при помощи команды **cat**:

```
$ cat H1.bin H2.bin | openssl dgst -sha3-256  
  
(stdin)=  
feb0524250ecc9498d51533eb74b793de052c7d01c8e8b9f021602d3980787  
dd
```

В результате корень дерева Меркла R будет равен значению хеша суммы узлов H1 и H2:

feb0524250ecc9498d51533eb74b793de052c7d01c8e8b9f021602d3980787dd.

Универсальное отображение хешей в шестнадцатеричном формате для узлов дерева Меркла H1 и H2 снова потребует использования команды **xxd** с параметром **-p**:

```
$ xxd -p H1.bin
```

```
db5e7494451556930405bfe2296a5578c6828af243afd726ef5d2ba4b2fb500
```

```
$ xxd -p H2.bin
```

```
677035391cd3701293d385f037ba32796252bb7ce180b00b582dd9b20aaad7f0
```

После всех вычислений построим само дерево Меркла, где в ячейках узлов для упрощения отобразим только первые 12 символов хешей (рис. 17).

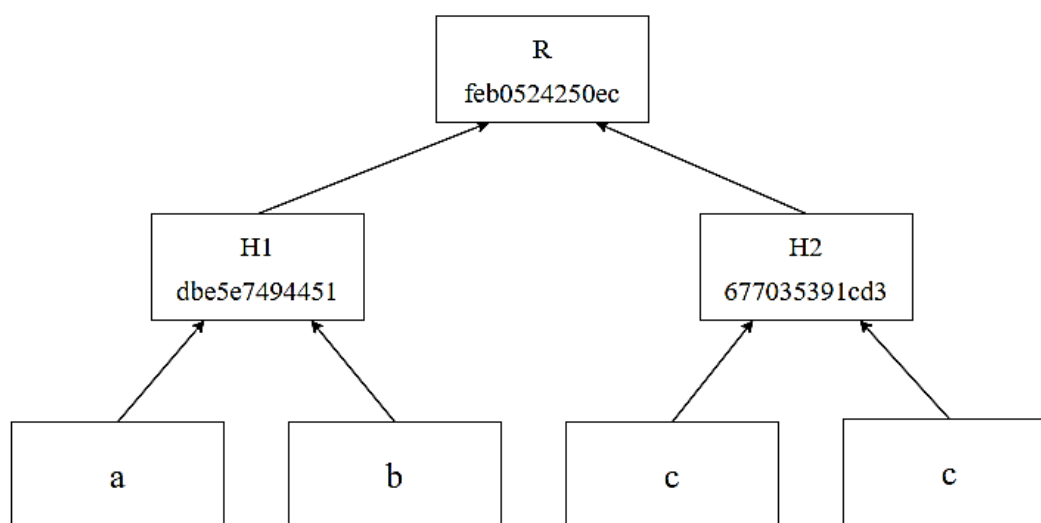


Рис. 17. Дерево Меркла

2.3.3. Доказательство Меркла

Использование деревьев Меркла в сети блокчейн позволяет не только сократить размер блоков, но и обеспечить упрощенную верификацию вхождения определенной транзакции в конкретный блок.

Так, в блоке *сжатого* блокчейна оказывается возможным хранение лишь корня дерева. В качестве хорошего показательного примера можно привести некоторое мобильное устройство, которое в силу своих недостаточно высоких аппаратно-программных характеристик зачастую просто

не способно хранить весь блокчейн целиком, поэтому содержит только корни деревьев дерева Меркла. Стационарному высокопроизводительному компьютеру, в свою очередь, ничто не мешает хранить полный блокчейн со всей информацией о транзакциях.

Проверка вхождения транзакции в блок осуществляется посредством доказательства Меркла. Данная процедура представляет собой процесс последовательной верификации ветви дерева от хеша исходной (проверяемой) транзакции до корня дерева. Идея доказательства Меркла состоит в том, что всегда существует некоторая ограниченная совокупность узлов дерева, на основе которых легко вычисляется значение хеша корня этого дерева при заданном значении хеша исходной транзакции.

Для примера вернемся к дереву Меркла, построенному ранее (рис. 16). Организуем процедуру верификации транзакции с хешем «а», т. е. докажем, что она действительно входит в состав данного дерева, относясь к конкретному блоку.

Для доказательства Меркла задается хеш проверяемой транзакции **а** и корень дерева **feb052...0787dd**. Вместе с тем процесс процедуры верификации транзакции, направленный от узла «а» к корню дерева последовательно, затрагивает узлы «b» и H2 (рис. 18).

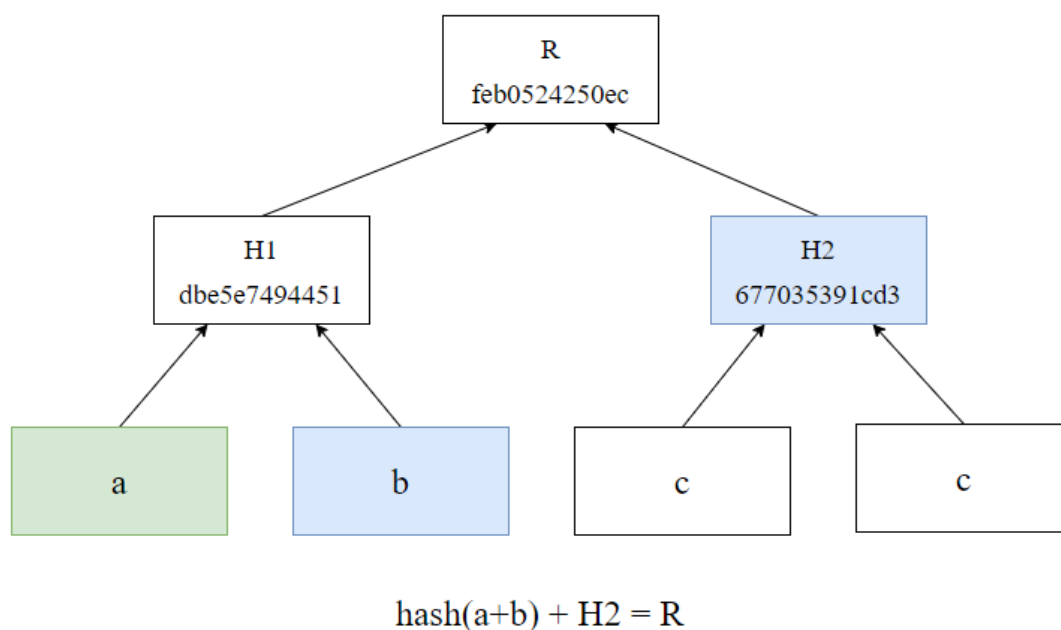


Рис. 18. Доказательство Меркла для транзакции с хешем «а»

Действительно, как видно из рис. 18, кроме хеша проверяемой транзакции и корня дерева в доказательстве Меркла принимают участие хеш **b** (для вычисления H1) и H2 (для вычисления R).

Рассмотрим также процедуру верификации транзакции T4 в построенном ранее более сложном дереве (рис. 19).

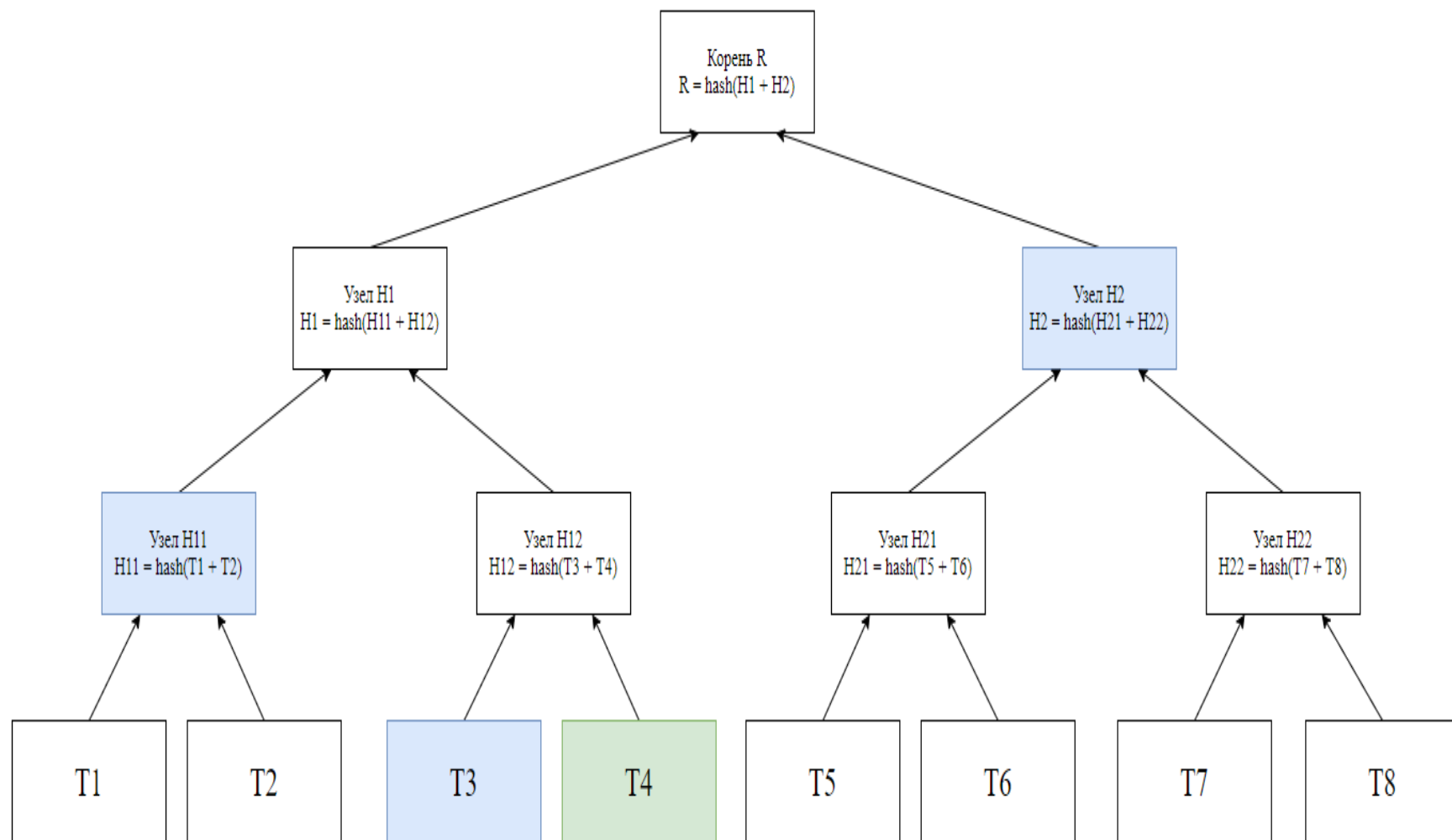


Рис. 19. Доказательство Меркла для транзакции T4

Очевидно (рис. 19), что в доказательстве Меркла принимают участие узлы, находящиеся в соседней ветке для каждого слоя дерева. Целью процедуры доказательства при этом является теоретическое значение корня дерева Меркла. Его совпадение с фактическим значением верифицирует транзакцию, подтверждая то, что она является узлом данного дерева и частью соответствующего блока.

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

- 1) разработать структуру транзакции в сети блокчейн для сферы деятельности из табл. 4;
- 2) создать три транзакции с описанной структурой, заполнив поля данными;
- 3) найти хеш-сумму для каждой транзакции, используя первый алгоритм хеширования из табл. 4;
- 4) найти хеш-сумму для каждой транзакции, используя второй алгоритм хеширования из табл. 4. Сравнить результаты хеширования со значениями предыдущего алгоритма хеширования;
- 5) используя любой из двух алгоритмов хеширования, построить дерево Меркла по трем транзакциям. Схематично изобразить дерево, указав на узлах первые пять символов полученных хешей;
- 6) сравнить в байтах размер данных при хранении в блоке только корня дерева Меркла, и при хранении всего списка транзакций;
- 7) по построенному дереву найти доказательство Меркла для любой из трех созданных транзакций.

Таблица 4

Варианты заданий согласно номеру в журнале обучающихся

№ вар.	Сфера применения блокчейна	Алгоритмы хеширования
0	Цифровое искусство	sha256, sha3_256
1	Недвижимость	md5, sha256
2	Сервисы для совместных поездок	sha3_512, md5
3	Стриминговые сервисы	sha224, sha3_512
4	Криптовалюта	sha384, sha224
5	Интернет вещей	sha512, sha384
6	Библиотеки	sha1, sha512
7	Кредитная история	sha3_256, sha1
8	Сервисы для размещения объявлений о продаже/покупке товаров	sha256, sha3_256
9	Единые государственные экзамены	md5, sha256

3. КРИПТОГРАФИЧЕСКИЕ СИСТЕМЫ

Описанные выше математические аппараты находят широкое применение в современных криптографических системах. Это касается протоколов как симметричного, так и асимметричного шифрования, а также механизмов цифровой подписи.

Здесь также затрагиваются вопросы целостности данных, рассматриваются области применения различных функций хеширования, в частности используемых для организации цифровой подписи и подтверждения подлинности транзакций в блокчейне.

Таким образом, цель настоящего пособия – не только ознакомить обучающихся с данной предметной областью, но и заинтересовать совершенно конкретными примерами реализации криптографических алгоритмов, вместе с тем предложив самостоятельно решить соответствующее контрольное задание в рамках изучаемой дисциплины.

3.1. Шифр Виженера и его реализация

Данный метод защиты информации получил свое название в честь французского дипломата XIX в. Блеза Виженера. Однако до этого на протяжении многих лет предлагались различные его модификации.

Шифр Виженера относится к категории методов полиалфавитного шифрования. Он считается симметричным и применим прежде всего для защиты буквенного текста. Общим ключом здесь, как правило, служит некоторое секретное слово [21].

Идея данного метода заключается в последовательном использовании более одного шифра Цезаря в зависимости от длины выбранного ключа. Классический шифр Цезаря, использованный, например, для русского алфавита, приведет к замене информационных символов по принципу фиксированного сдвига. Так, если сдвиг составляет, к примеру, «+4», то все буквы «А» в тексте будут заменены на буквы «Д», все буквы «Б» – на буквы «Е» и т. д.

Секретное слово, выполняющее роль ключа в шифре Виженера, вносит определенную степень случайности в процесс данного сдвига. Очевидно, что чем длинней ключ и чем он менее осмысленный, тем более криптостойкой окажется данная система передачи информации и больше времени понадобится криптоаналитику для анализа перехваченного текста.

Пусть n – число символов алфавита, m_i – символ открытого текста на i -й позиции информационного сообщения, k_j – символ ключа на j -й позиции ключа. Тогда процесс шифрования при использовании шифра Виженера,

подобно процедуре сложения элементов аддитивной группы, можно представить следующим образом:

$$c_i = (m_i + k_j) \bmod n. \quad (31)$$

Процесс расшифрования, соответственно, будет иметь вид:

$$m_i = (c_i + n - k_j) \bmod n. \quad (32)$$

	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
А	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Б	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
В	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б
Г	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В
Д	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г
Е	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д
Ё	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е
Ж	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё
З	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж
Й	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З
И	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й
К	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И
Л	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К
М	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л
Н	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М
О	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н
П	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О
Р	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П
С	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р
Т	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С
У	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т
Ф	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У
Х	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ъ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ы	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы
Э	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь
Ю	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Рис. 20. Таблица Виженера

Теоретически i может не иметь ограничений в силу непрерывного потока информационных элементов, в то время как длина ключа является фиксированной. Поэтому для новых информационных символов m_i вполне допустимы повторяющиеся значения символов ключа k_j .

Данная ситуация наглядно демонстрирует очевидное снижение криптостойкости при использовании коротких ключей, а также показывает сводимость шифра Виженера к шифру Цезаря в том случае, когда ключ будет состоять лишь из одного символа. Для работы с шифром Виженера целесообразно использовать построенную заранее таблицу алфавита или таблицу Виженера. Каждая ее строка и столбец содержат полный алфавит и формируются по принципу циклического сдвига.

В случае с русским алфавитом таблица Виженера имеет размерность 33×33 символа (рис. 20).

В результате каждый информационный символ сообщения на русском языке в зависимости от используемого ключа путем гаммирования по правилу (31) может быть преобразован в один из 33 других символов русского алфавита.

Построенная таблица (рис. 20) приводит в пересечении строки и столбца результат выполнения процедуры (31) для их заглавных символов. То есть если, например, требуется зашифровать информационный символ $m_i = \langle \text{Д} \rangle$ некоторым ключом, у которого $k_j = \langle \text{М} \rangle$, то результатом пересечения соответствующих строки и столбца будет символ $c_i = \langle \text{Р} \rangle$ (рис. 21).

	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	...
А	А	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	...
Б	Б	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	...
В	В	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	...
Г	Г	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	...
Д	Д	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	...
Е	Е	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	...
Ё	Ё	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	...
Ж	Ж	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	...
З	З	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	...
Й	Й	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	...
И	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	...
К	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	...
Л	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	...
М	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	...
Н	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Рис. 21. Результат выполнения процедуры (31)

Обратный процесс расшифрования по правилу (32) использует аналогичное обращение к таблице Виженера (рис. 20). Легко убедиться, что искомым символом $m_i = \langle \text{Д} \rangle$, так же как и $k_j = \langle \text{М} \rangle$, останется заглавным символом

строки или столбца, в то время как $c_i = \langle P \rangle$ продолжит находиться на их пересечении.

Первая практическая работа приведена здесь в качестве разминочной. Но стоит отметить, что, несмотря на кажущуюся примитивность, шифр Виженера, в частности его продвинутые модификации, и по сегодняшний день представляет проблему для простых методов криптоанализа. Это касается, например, шифра Вернама, разработанного еще в 1917 г. и являющегося наглядным примером системы с абсолютной криптографической стойкостью.

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания:

- 1) выбрать на свое усмотрение произвольное слово или фразу для формирования ключа;
- 2) используя таблицу Виженера, зашифровать свои данные: фамилию, имя, отчество;
- 3) используя таблицу Виженера, выполнить проверку, расшифровав полученное сообщение.

По результатам выполнения задания должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

3.2. Метод Диффи – Хеллмана

Протокол Диффи – Хеллмана (DH – Diffie–Hellman key exchange protocol), разработанный еще в 1976 г., следовало бы отнести к категории асимметричных алгоритмов шифрования, однако на практике он применяется только для формирования общего секретного ключа, который планируется в дальнейшем использовать в алгоритмах симметричного шифрования [22].

Сегодня DH находит применение в рамках организации различных TLS-соединений, для построения виртуальных частных сетей, в частности OpenVPN, а также при рукопожатиях в сетях Wi-Fi с WPA3.

Безопасность DH основана на сложности вычисления дискретного логарифма в конечном поле $GF(p)$ (7), где в качестве p задается достаточно большое число, как правило его разрядность превышает 1024 бит. Проблема дискретного логарифмирования была затронута в подразделе 1.2.4.

3.2.1. Схема Диффи – Хеллмана для двух абонентов

Схематично метод формирования общего секретного ключа по принципу ДН для двух сторон представлен на рис. 22.

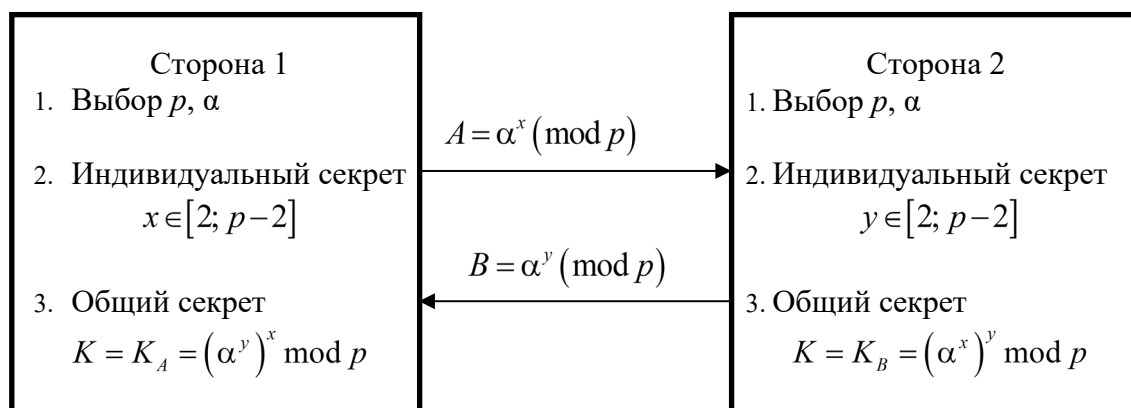


Рис. 22. Алгоритм Диффи – Хеллмана для формирования общего ключа

На первом шаге стороны определяют характеристику p конечного поля $GF(p)$, которое должно быть большим простым числом, а также выбирают первообразный элемент α , который в соответствии с теорией должен удовлетворять правилу (3).

Вторым шагом стороны выбирают индивидуальные секретные компоненты x и y , которые представляют собой случайные большие числа из диапазона $[2; p-2]$. Связано это с тем, что крайние значения 1 и $(p-1)$ изначально бессмысленны в качестве секретных компонент. Число 1 при возведении в любую степень останется прежним, $(p-1)$ в силу тождества (3) также приведет к результату 1.

Выбранные числа будут степенями первообразного элемента: α^x и α^y . Таким образом, формируются открытые компоненты $A = \alpha^x$ и $B = \alpha^y$, обмен которыми осуществляется по небезопасному каналу связи.

На третьем шаге стороны вычисляют значение общего секретного ключа K , который впоследствии будет использован для симметричного шифрования. Хотя ничто не мешает применять его также в качестве асимметричного закрытого ключа по отношению к одному из открытых ключей, например A или B . Однако, как было отмечено выше, такой режим работы ДН в настоящее время не используется.

Приведем простейший пример формирования общего секретного ключа по принципу ДН для некоторого поля $GF(7)$. В подразделе 1.2.1 было доказано, что в этом поле будет как минимум один первообразный элемент, и это число 3. Таким образом, оба стартовых параметра известны: $p = 7$, $\alpha = 3$. В качестве секретных компонент на стороне 1 и 2 выберем случайным образом числа, например $x = 3$ и $y = 5$.

Следуя алгоритму, стороны возводят первообразный элемент в степени, соответствующие секретным компонентам:

$$A = \alpha^x = 3^3 = 27 = 6 \pmod{7};$$

$$B = \alpha^y = 3^5 = 243 = 5 \pmod{7}.$$

Обмен открытыми компонентами A и B , как уже было выше сказано, производится по небезопасным каналам и теоретически может быть перехвачен криптоаналитиком. Однако сложность процедуры нахождения дискретного логарифма (7) при достаточно большом значении p заставляет считать данный метод вполне криптостойким.

Обменявшись открытыми компонентами, стороны приступают к формированию общего секретного ключа, который будет использоваться в рамках текущего сеанса связи:

$$K_A = (\alpha^y)^x = B^x = 5^3 = 125 = 6 \pmod{7};$$

$$K_B = (\alpha^x)^y = A^y = 6^5 = 7776 = 6 \pmod{7};$$

$$K = K_A = K_B.$$

3.2.2. Схема Диффи – Хеллмана для множества абонентов

Данный метод формирования общего ключа может быть использован и для большего числа абонентов. Логический смысл схемы (рис. 22) при этом не меняется. Все потенциальные обладатели общего секретного ключа, по сути, должны последовательно обмениваться между собой открытыми компонентами. Реализовать такой алгоритм можно при помощи итерационной схемы, где число итераций будет на единицу меньше числа абонентов. Приведем пример с тремя абонентами (рис. 23 и 24).

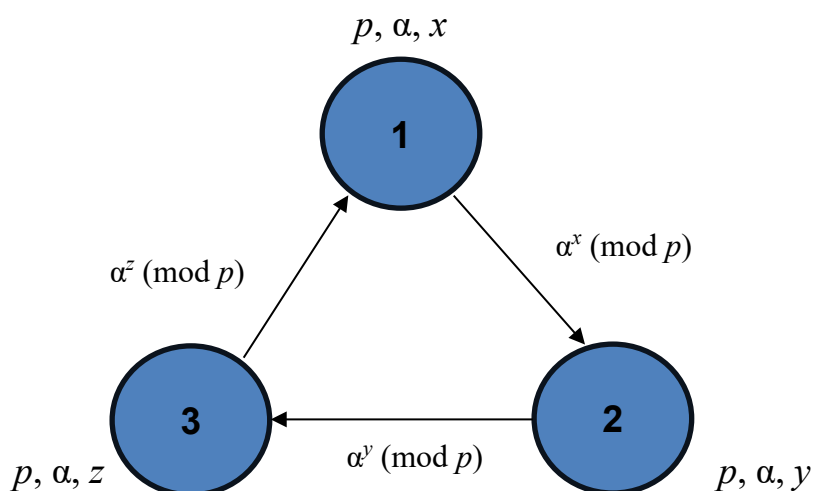


Рис. 23. Первая итерация ДН для трех абонентов

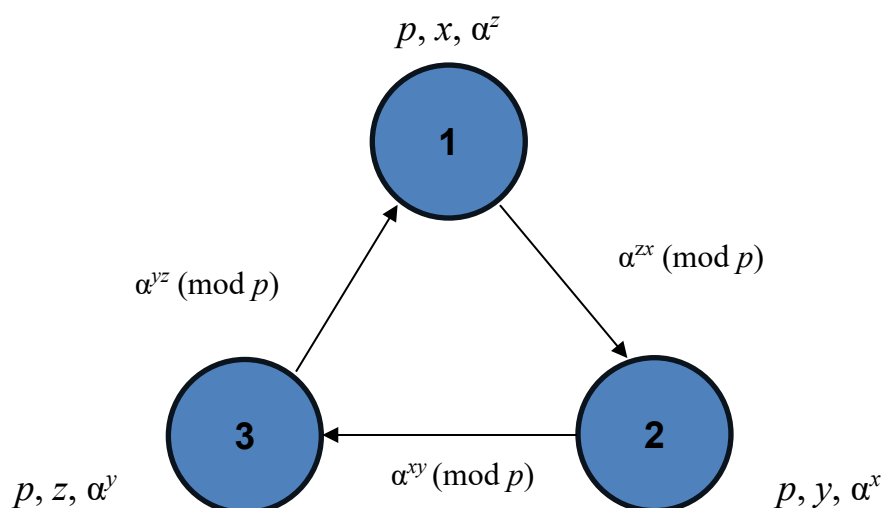


Рис. 24. Вторая итерация ДН для трех абонентов

После второй итерации каждый абонент будет иметь необходимую информацию для формирования общего секретного ключа:

$$\begin{array}{ll}
 \text{абонент 1: } p, x, \alpha^{yz}; & K_1 = (\alpha^{yz})^x = \alpha^{yzx} \pmod{p}; \\
 \text{абонент 2: } p, y, \alpha^{zx}; & K_2 = (\alpha^{zx})^y = \alpha^{zxy} \pmod{p}; \\
 \text{абонент 3: } p, z, \alpha^{xy}; & K_3 = (\alpha^{xy})^z = \alpha^{xyz} \pmod{p}.
 \end{array}$$

Очевидным является равенство индивидуально сформированных абонентами ключей: $K = K_1 = K_2 = K_3$.

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

1) реализовать схему формирования общего секретного ключа для двух абонентов:

- а) выбрать из табл. 5 значения параметров p, x, y ,
- б) определить первообразный элемент поля $GF(p)$,
- в) построить схему обмена открытыми компонентами между двумя абонентами,

г) сформировать общий секретный ключ для двух абонентов;

2) реализовать схему формирования общего секретного ключа для трех абонентов:

- а) выбрать из табл. 5 значения параметров p, x, y, z ,

- б) построить схемы обмена открытыми компонентами между тремя абонентами,
в) сформировать общий секретный ключ для трех абонентов.

Таблица 5

Варианты заданий согласно номеру в журнале обучающихся

№ варианта	Характеристика поля p	Секретные компоненты			№ варианта	Характеристика поля p	Секретные компоненты		
		x	y	z			x	y	z
1	17	5	6	7	16	11	5	6	7
2	19	3	5	7	17	17	12	11	14
3	31	11	15	4	18	23	5	12	14
4	37	26	16	18	19	19	16	2	5
5	11	9	3	4	20	31	4	3	18
6	41	4	2	7	21	37	20	18	11
7	29	15	19	20	22	41	15	7	21
8	23	11	15	6	23	11	2	8	7
9	19	12	4	3	24	29	16	17	26
10	17	3	8	9	25	17	3	15	12
11	11	8	9	4	26	37	6	5	27
12	37	30	32	34	27	19	8	11	17
13	29	4	7	9	28	11	6	8	10
14	31	15	17	11	29	17	12	14	9
15	19	8	9	10	30	23	4	6	8

3.3. Схема разделения секрета Шамира

Данный алгоритм был разработан в 1979 г. Шамиром и позволяет реализовать (k, n) -пороговое разделение секретного сообщения между n сторонами таким образом, чтобы только k из них могли его восстановить [23].

Стоит отметить важные моменты данного алгоритма:

- для восстановления секрета достаточно k любых сторон;
- никакие $(k - 1)$ и менее сторон не смогут получить доступ к секрету.

В основе схемы Шамира лежит теорема, доказывающая существование единственного интерполяционного полинома для заданного на некотором отрезке набора точек [24].

Пусть имеются точки с координатами (x_i, y_i) , где $i = 0 \dots (k-1)$. При этом все x_i различны. Существует единственный многочлен степени не выше k , для которого будет справедливо равенство $L(x_i) = y_i$:

$$L(x) = \sum_{j=0}^k y_j l_j(x). \quad (33)$$

Многочлен вида (33) получил название интерполяционного полинома Лагранжа. Входящие в его состав базисные полиномы вычисляются по формуле:

$$l_j(x) = \prod_{i=0, i \neq j}^k \frac{(x - x_i)}{(x_j - x_i)} = \frac{(x - x_0)}{(x_j - x_0)} \dots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \cdot \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \dots \frac{(x - x_k)}{(x_j - x_k)}. \quad (34)$$

Таким образом, для интерполяции полинома степени $k-1$ требуется k точек. При меньшем числе точек интерполяция станет невозможной. В этом и заключается основная идея схемы Шамира. Коэффициентами интерполяционного полинома в данном случае будут элементы поля $GF(p)$.

Очевидно, что схема разделения секрета Шамира находит применение для решения задач многопользовательской авторизации.

3.3.1. Процедура разделения и восстановления секрета

Пусть имеется некоторый секрет M , который требуется разделить между n сторонами. При этом любые k из них должны иметь возможность его восстановить. Такая формулировка задачи равносильна реализации (k, n) -пороговой схемы.

1. Определим простое число $p > \max(M, n)$, по которому можно построить поле $GF(p)$.

2. Над выбранным полем $GF(p)$ построим многочлен степени $k-1$, все коэффициенты которого будут выбраны случайно, кроме $a_0 = M$:

$$F(x) = \sum_{j=k-1}^{j=0} a_j x^j \bmod p = (a_{k-1} x^{k-1} + a_{k-2} x^{k-2} + \dots + a_1 x + M) \bmod p. \quad (35)$$

Таким образом, полином (35) будет содержать разделяемый секрет, а на месте остальных коэффициентов будут случайные числа, при этом должно выполняться общее условие:

$$0 \leq a_j \leq p-1. \quad (36)$$

3. Вычислим значения многочлена (35) в так называемых точках «тени»:

$$\sigma_i = F(i) \bmod p, \quad 1 \leq i \leq n. \quad (37)$$

Подставляя различные значения аргумента в (37), получим n значений σ_i :

$$\begin{aligned} \sigma_1 &= F(1) = (a_{k-1} \cdot 1^{k-1} + a_{k-2} \cdot 1^{k-2} + \dots + a_1 \cdot 1 + M) \bmod p, \\ \sigma_2 &= F(2) = (a_{k-1} \cdot 2^{k-1} + a_{k-2} \cdot 2^{k-2} + \dots + a_1 \cdot 2 + M) \bmod p, \\ &\dots\dots\dots \\ \sigma_i &= F(i) = (a_{k-1} \cdot i^{k-1} + a_{k-2} \cdot i^{k-2} + \dots + a_1 \cdot i + M) \bmod p, \\ &\dots\dots\dots \\ \sigma_n &= F(n) = (a_{k-1} \cdot n^{k-1} + a_{k-2} \cdot n^{k-2} + \dots + a_1 \cdot n + M) \bmod p. \end{aligned}$$

4. Каждая сторона, принимающая участие в разделении секрета M , должна получить свою долю секрета σ_i или так называемую «тень». При этом всем сторонам становится известна степень полинома $k - 1$ и характеристика поля p .

5. Все коэффициенты a_j и секрет M «забываются».

Известно, что для восстановления секрета M необходимо собрать как минимум k теней σ_i . Только в этом случае станет возможным вычислить коэффициенты a_j многочлена (35), и в том числе $a_0 = M$.

Таким образом, объединив свои тени σ_i , которые фактически соответствуют точкам $(x_i, y_i) = (i, \sigma_i)$, стороны представляют $F(x)$ в форме полинома Лагранжа (33):

$$F(x) = \sum_{i=1}^k l_j(x) y_i \bmod p, \quad (38)$$

который также состоит из базисных полиномов (34):

$$l_j(x) = \prod_{j \neq i} \frac{(x - x_i)}{(x_j - x_i)} \bmod p, \quad (39)$$

где $(x_i, y_i) = (i, \sigma_i)$ – координаты точек $F(x)$.

3.3.2. Реализация схемы Шамира

Реализуем (k, n) -пороговое разделение секретного сообщения по схеме Шамира с параметрами $n = 5, k = 3$. Сам секрет при этом будет иметь значение $M = 11$.

Выберем $p = 59$, которое гарантированно будет удовлетворять условию $p > \max(11, 5)$.

Определим в общем виде многочлен $F(x)$ степени $k - 1 = 3 - 1 = 2$:

$$F(x) = (a_2x^2 + a_1x + a_0) \bmod p.$$

Коэффициенты a_j должны удовлетворять условию (36), при этом $a_0 = M$, a_1 и a_2 – случайные числа. В соответствии с данным алгоритмом они должны быть забыты после распределения долей.

Пусть $a_1 = 23, a_2 = 10$.

Тогда $F(x)$ принимает вид:

$$F(x) = (10x^2 + 23x + 11) \bmod 59.$$

Определим тени:

$$\sigma_1 = F(1) = (10 \cdot 1^2 + 23 \cdot 1 + 11) = 44 \bmod 59;$$

$$\sigma_2 = F(2) = (10 \cdot 2^2 + 23 \cdot 2 + 11) = 97 = 38 \bmod 59;$$

$$\sigma_3 = F(3) = (10 \cdot 3^2 + 23 \cdot 3 + 11) = 170 = 52 \bmod 59;$$

$$\sigma_4 = F(4) = (10 \cdot 4^2 + 23 \cdot 4 + 11) = 263 = 27 \bmod 59;$$

$$\sigma_5 = F(5) = (10 \cdot 5^2 + 23 \cdot 5 + 11) = 376 = 22 \bmod 59.$$

Каждый носитель доли секрета получает свою тень (i, σ_i) , а также значение $p = 59$. Таким образом, среди всех пяти участников распределяются точки:

$$(x_1, y_1) = (1, 44);$$

$$(x_2, y_2) = (2, 38);$$

$$(x_3, y_3) = (3, 52);$$

$$(x_4, y_4) = (4, 27);$$

$$(x_5, y_5) = (5, 22).$$

Для гарантированного восстановления секрета требуется собрать только три любые точки, например: (x_2, y_2) , (x_3, y_3) и (x_5, y_5) .

Определим базисные полиномы на основании выражения (39):

$$\begin{aligned}l_2(x) &= \frac{x-3}{2-3} \cdot \frac{x-5}{2-5} = \frac{1}{3}(x^2 - 8x + 15); \\l_3(x) &= \frac{x-2}{3-2} \cdot \frac{x-5}{3-5} = -\frac{1}{2}(x^2 - 7x + 10); \\l_5(x) &= \frac{x-2}{5-2} \cdot \frac{x-3}{5-3} = \frac{1}{6}(x^2 - 5x + 6).\end{aligned}$$

Построим интерполяционный полином Лагранжа по правилу (38):

$$F(x) = \left[\frac{y_2}{3}(x^2 - 8x + 15) - \frac{y_3}{2}(x^2 - 7x + 10) + \frac{y_5}{6}(x^2 - 5x + 6) \right] \bmod 59.$$

Путем математических преобразований приведем его к виду:

$$F(x) = \left[\frac{1}{6}(-58x^2 + 374x - 288) \right] \bmod 59 = 6^{-1}(-58x^2 + 374x - 288) \bmod 59.$$

Для нахождения обратного элемента по умножению для числа 6 в поле GF(59), как вариант, можно воспользоваться расширенным алгоритмом Евклида.

Не составляет труда свести уравнение (6) к виду:

$$\alpha x + py = 1, \quad (40)$$

чтобы впоследствии решить его при помощи алгоритма, описанного в подразделе 1.2.3.

Поскольку единица может делиться только на единицу, то данное уравнение имеет решение только в случае, если $\text{НОД}(\alpha, p) = 1$.

Взяв от обеих частей уравнения остаток по модулю p , получим:

$$\alpha x = 1 \bmod p.$$

Таким образом, в соответствии с (5) можно сделать вывод, что $x = \alpha^{-1}$ и уравнение (40) позволит его определить при помощи расширенного алгоритма Евклида.

В данном примере обратным элементом для числа 6 в поле GF(59) является число 10. Это подтверждается тем же тождеством (5):

$$6 \cdot 10 = 60 \equiv 1 \pmod{59}.$$

Учитывая это, полином Лагранжа принимает вид:

$$F(x) = [10(-58x^2 + 374x - 288)] \bmod 59 = (-580x^2 + 3740x - 2880) \bmod 59.$$

Для получения секретного сообщения достаточно выделить только коэффициент a_0 :

$$M = a_0 = -2880 = 11 \pmod{59}.$$

Однако попутно можно определить и другие коэффициенты, упростив выражение для интерполяционного полинома Лагранжа:

$$F(x) = (10x^2 + 23x - 11) \bmod 59.$$

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется организовать (3, 5)-пороговое разделение секретного сообщения при помощи схемы Шамира. Для этого необходимо выполнить следующие действия:

- 1) выбрать из табл. 6 значения параметров характеристику поля p , значение секрета M и коэффициенты a_j для построения полинома Лагранжа;
- 2) распределить доли между хранителями секрета;
- 3) произвести восстановление секрета для любых трех долей. Убедиться в успехе данной операции;
- 4) произвести восстановление секрета для любых двух долей. Убедиться в безуспешности данной операции;
- 5) произвести восстановление секрета для любых четырех долей. Убедиться в успехе данной операции.

Таблица 6

Варианты заданий согласно номеру
в журнале обучающихся

№ варианта	p	M	a_1	a_2	№ варианта	p	M	a_1	a_2
1	37	5	16	19	16	41	30	29	28
2	41	9	7	8	17	43	26	41	5
3	53	30	32	45	18	59	55	6	14
4	47	32	6	26	19	47	40	4	5
5	67	54	2	4	20	61	2	18	12

№ варианта	p	M	a_1	a_2	№ варианта	p	M	a_1	a_2
6	61	3	17	19	21	37	3	11	34
7	53	31	5	3	22	53	32	7	9
8	47	15	16	34	23	41	24	6	5
9	59	11	47	2	24	37	15	14	15
10	61	24	34	15	25	43	18	16	19
11	43	34	17	17	26	61	20	5	3
12	67	50	4	7	27	67	42	8	6
13	41	6	12	12	28	59	16	21	32
14	37	12	6	8	29	41	19	35	27
15	53	14	9	3	30	43	15	12	11

3.4. Схема Эль-Гамала

Данный алгоритм защиты информации, разработанный Тахером Эль-Гамалем в 1984 г., является развитием алгоритма Диффи – Хеллмана и обеспечивает решение двух основных криптографических задач, связанных с использованием асимметричных ключей: непосредственно шифрования и аутентификации [25].

Как и в случае с методом Диффи – Хеллмана, криптографическая стойкость данного алгоритма будет зависеть от сложности определения дискретного логарифма в конечном поле.

На практике схема Эль-Гамала применяется чаще всего для формирования цифровой подписи в соответствии с алгоритмом DSA.

3.4.1. Цифровая подпись на основе схемы Эль-Гамала

Основной задачей цифровой подписи является аутентификация отправителя сообщения. Согласно алгоритму, сторона A , подтверждающая свою подлинность, формирует два ключа: O_A (открытый) и S_A (закрытый), которые связаны между собой, подобно тому, как это было в алгоритме Диффи – Хеллмана (рис. 25).

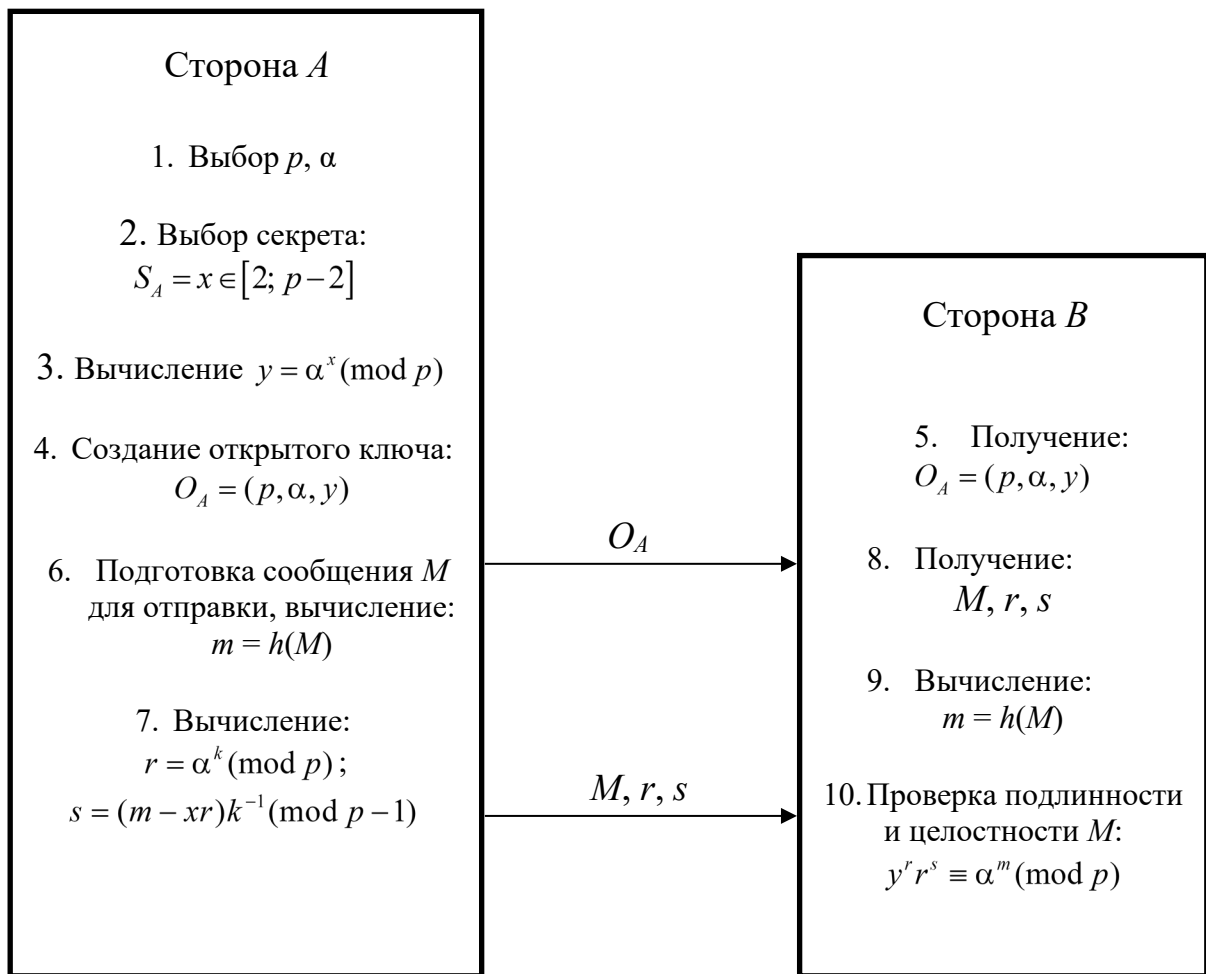


Рис. 25. Алгоритм цифровой подписи
на основе схемы Эль-Гамала

Закрытый ключ S_A выбирается из множества всех элементов конечного поля $\text{GF}(p)$ с первообразным корнем α , где p – простое число. Открытый ключ представляет собой набор (p, α, y) , где $y = \alpha^x \pmod{p}$.

Ключ O_A в соответствии со схемой (рис. 25) публикуется и становится доступным для всех, кто планирует проводить проверку подлинности стороны A . Таким образом, O_A передается стороне B по открытому каналу связи.

Пусть сторона A планирует отправку некоторого сообщения M стороне B . При этом стороне A требуется решить две задачи: защитить M от подмены и доказать стороне B свое авторство по отношению к данному сообщению.

Защита подлинности сообщения эффективно решается при помощи некоторой функции хеширования $h(M)$. Таким образом, формируется дайджест сообщения M : $m = h(M)$.

На следующем шаге выбирается некоторое случайное число k , которое удовлетворяет условию $1 < k < (p-1)$ и при этом является взаимно простым с $(p-1)$.

Сначала вычисляется первый компонент цифровой подписи сообщения:

$$r = \alpha^k \pmod{p}. \quad (41)$$

Второй компонент подписи определяется по формуле:

$$s = (m - xr)k^{-1} \pmod{p-1}. \quad (42)$$

Таким образом, выражения (41) и (42) позволяют сформировать пару (r, s) , которая будет выступать в роли цифровой подписи для сообщения M . При этом параметры подписи r и s должны удовлетворять условиям:

$$\begin{aligned} 0 < r < p; \\ 0 < s < p-1. \end{aligned} \quad (43)$$

Проверка подписи принятого стороной B сообщения M производится при помощи компонентов O_A и непосредственно параметров самой подписи. Так, например, если не выполняется хотя бы одно из условий (43), то подпись считается недействительной.

В остальном сторона B также вычисляет дайджест принятого сообщения $m = h(M)$, и подпись считается правильной в том случае, если выполняется тождество:

$$y^r r^s \equiv \alpha^m \pmod{p}. \quad (44)$$

Приведем пример. Пусть имеется некоторое сообщение $M = \text{text}$, которое требуется подписать на стороне A , передать вместе с цифровой подписью стороне B и проверить подлинность подписи.

Приступим к формированию O_A , выбрав подходящее поле $\text{GF}(p)$ с параметрами $p = 23$ и $\alpha = 5$. Зададим секретный ключ $S_A = x$, который будет удовлетворять условию $1 < x < p-1$, например, $S_A = 9$.

Определим параметр $y = \alpha^x = 5^9 = 1953125 = 11 \pmod{23}$. Теперь открытый ключ $O_A = (p, \alpha, y) = (23, 5, 11)$ полностью сформирован и может быть опубликован.

Вычислим хеш-функцию $h(M)$. В данной задаче качество хеш-функции не так важно, поэтому будем считать результат ее вычисления произвольным, например $m = h(M) = h(\text{text}) = 12$.

Выберем случайное k , которое должно удовлетворять условию $1 < k < 22$ и являться взаимно простым с числом 22, например $k = 7$.

Используя формулы (41) и (42), вычислим компоненты цифровой подписи:

$$r = 5^k = 5^7 = 78125 = 17(\bmod 23);$$

$$s = (m - xr)k^{-1} = (12 - 9 \cdot 17) \cdot 7^{-1} = (-141) \cdot 19 = 5(\bmod 22).$$

Таким образом, сторона A отправляет стороне B сообщение $M = \text{text}$, подписанное цифровой подписью $(r, s) = (17, 5)$.

Получив данные $(M, r, s) = (\text{text}, 17, 5)$, сторона B осуществляет проверку их подлинности.

Для этого вычисляется хеш-функция $h(M)$, которая будет иметь значение $m = h(\text{text}) = 12$.

Затем в соответствии с выражением (44) подтверждается тождество, которое говорит о том, что подпись является подлинной:

$$11^{17} \cdot 17^5 \equiv 5^{12}(\bmod 23);$$

$$14 \cdot 21 \equiv 18(\bmod 23);$$

$$294 \equiv 18(\bmod 23).$$

3.4.2. Шифрование с открытым ключом по схеме Эль-Гамала

Схематично процесс шифрования на основе схемы Эль-Гамала с использованием асимметричных ключей представлен на рис. 26.

Ключи S_A и O_A определяются при помощи того же алгоритма, что и для цифровой подписи. Секрет $S_A = x$ выбирается из множества всех элементов поля $GF(p)$ с первообразным корнем α , где p – простое число. Открытый ключ будет содержать набор $O_A = (p, \alpha, y)$, где $y = \alpha^x (\bmod p)$.

Пусть сторона A должна передать стороне B некоторое зашифрованное сообщение M , которое должно быть элементом поля $GF(p)$. Тогда обладателем закрытого ключа S_A будет сторона B , так как только она является адресатом зашифрованного сообщения. Таким образом, именно сторона B должна стать инициатором формирования ключей или получить S_A из доверительного источника по безопасному каналу связи.

Процедура шифрования предполагает вычисление пары чисел:

$$\begin{aligned} m_1 &= \alpha^k (\bmod p); \\ m_2 &= y^k M (\bmod p), \end{aligned} \tag{45}$$

где параметр k представляет собой некоторое случайное целое число, которое должно удовлетворять условию: $1 < k < p - 1$. При этом имеет смысл переопределять k для каждого отдельно взятого блока шифруемых данных.

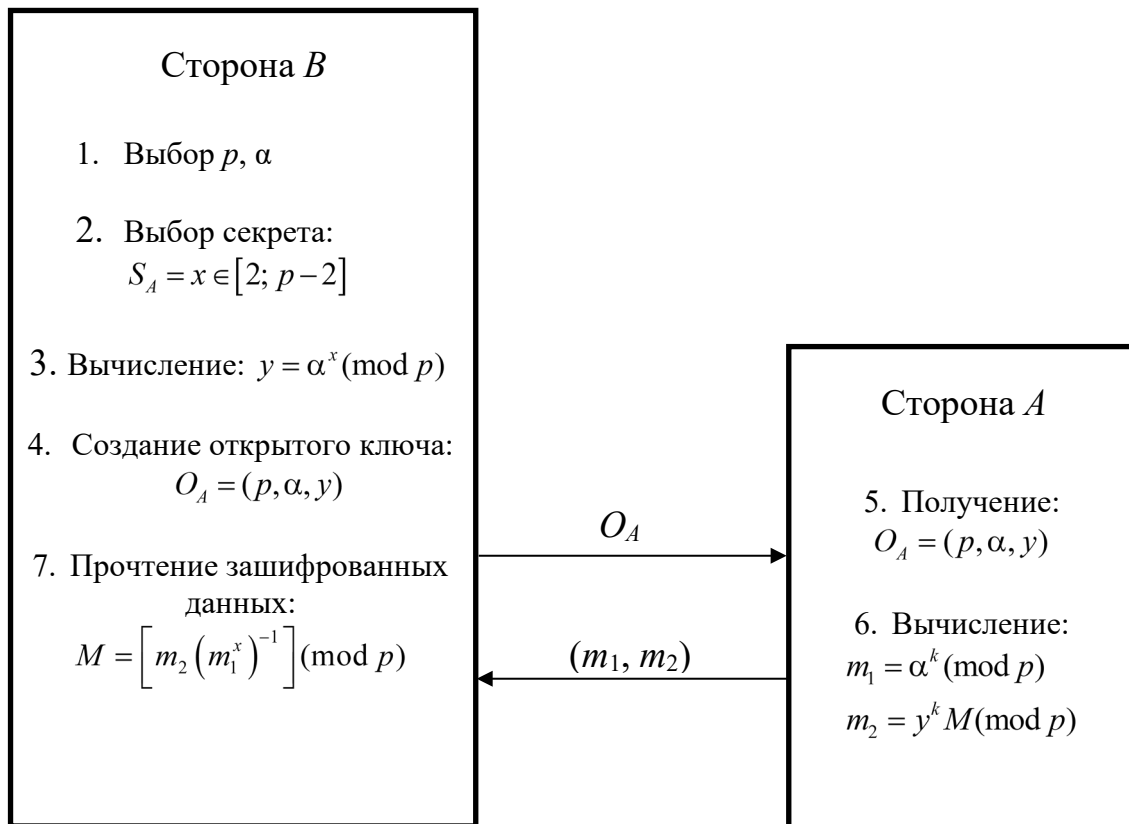


Рис. 26. Алгоритм шифрования
на основе схемы Эль-Гамала

Пара чисел (m_1, m_2) является зашифрованным сообщением и передается стороне B по открытому каналу связи.

На приемной стороне B производится прочтение зашифрованных данных:

$$M = \left[m_2 \left(m_1^x \right)^{-1} \right] \pmod p. \quad (46)$$

Приведем пример. Пусть сторона B должна принять некоторое сообщение $M = 7$ от стороны A . Тогда сторона B должна сформировать ключ S_A и O_A . И при этом ключ O_A требуется передать стороне B .

Таким образом, сторона B выбирает некоторое простое число p , над которым можно построить такое поле $\text{GF}(p)$, что сообщение M будет его элементом. Пусть $p = 11$, тогда над ним можно построить поле $\text{GF}(11)$ с первообразным элементом $\alpha = 2$.

Третий компонент открытого ключа y определяется на основе закрытого ключа $S_A = x$, который выбирается случайным образом из этого же поля $GF(11)$. Пусть $x = 6$, тогда $y = 2^6 = 64 = 9 \pmod{11}$.

Сторона B сохраняет закрытый ключ S_A в секрете, открытый ключ $O_A = (p, \alpha, y) = (11, 2, 9)$ публикует, предоставляя доступ к нему стороне A .

Сторона A , воспользовавшись ключом O_A , производит вычисление пары чисел (45), выбрав при этом некоторое случайное значение $k = 3$, удовлетворяющее условию $1 < k < 10$:

$$m_1 = 2^3 = 8 \pmod{11};$$

$$m_2 = 9^3 \cdot 7 = 5103 = 10 \pmod{11}.$$

Пара чисел $(m_1, m_2) = (8, 10)$ передается стороне B , которая, обладая закрытым ключом S_A может восстановить исходное сообщение при помощи операции (46):

$$M = \left[10 \cdot (8^6)^{-1} \right] = 10 \cdot 3^{-1} = 10 \cdot 4 = 40 = 7 \pmod{11}.$$

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

1) подписать сообщение M_1 из табл. 7 цифровой подписью по схеме Эль-Гамала. В качестве хеш-функции использовать сумму ASCII кодов сообщения M_1 , сжав его до одного байта. Ключи сформировать самостоятельно на основе значений p и x из табл. 7. Произвести проверку подлинности цифровой подписи сообщения M_1 ;

2) осуществить шифрование сообщения M_2 из табл. 7 по схеме Эль-Гамала с открытым ключом. Сформировать ключи на основе значений p и x из табл. 7. Зашифровать сообщение M_2 и расшифровать его.

Таблица 7

Варианты заданий согласно номеру в журнале обучающихся

№ варианта	M_1	p	x	M_2	№ варианта	M_1	p	x	M_2
1	Abcd	37	12	16	16	http	37	21	15
2	12345\$	31	17	20	17	Рфзтнбв	31	15	18
3	qWeRtY	43	15	11	18	gsgsgsgs	43	19	26

№ варианта	M_1	p	x	M_2	№ варианта	M_1	p	x	M_2
4	Текст	29	9	7	19	1a1a1a1a	29	28	12
5	111111	41	19	35	20	0b0b0b0b	41	35	28
6	cfcfcfcf	29	7	4	21	Urwk195	43	16	11
7	000000	37	32	21	22	19293949	29	12	19
8	5vjslas	31	11	24	23	Ntrsad85	37	5	24
9	Нaxsq	43	24	4	24	564#&12	29	24	14
10	ddddgggg	37	16	17	25	*1@&*&	41	4	26
11	96tdpad	31	14	15	26	##%%	37	13	18
12	ржвефл	41	18	25	27	D#%&	31	12	15
13	Втфьюя	29	2	19	28	4f&@7	43	29	12
14	ммкфАР	37	4	20	29	3862&	29	16	18
15	лыжгйд	41	26	18	30	6fr	41	24	26

3.5. Криптосистема RSA

Алгоритм разработан в 1976 г. и назван в честь трех своих создателей: Rivest, Shamir и Adleman (Рональд Линн Ривест, Ади Шамир, Леонард Макс Адлеман). Аналогично схеме Эль-Гамала данная система использует асимметричные ключи, решая задачи цифровой подписи и шифрования с открытым ключом [26].

Алгоритм RSA используется в большом числе криптографических приложений, включая PGP, S/MIME, TLS/SSL, IPSEC/IKE.

3.5.1. Создание ключей RSA

Надежность данной криптографической системы обеспечивается сложностью задачи факторизации произведения двух больших простых чисел. Базовой операцией в алгоритме RSA является возведение в степень по модулю большого числа.

В криптосистеме RSA открытый и закрытый ключи представляют собой пару целых чисел. При этом они являются взаимно обратными. Это означает, что с математической точки зрения не имеет значения, какой ключ будет выступать в роли открытого, а какой будет секретным.

Процедура формирования ключей в RSA опирается на теорему Эйлера с учетом того, что аргумент n функции Эйлера $\phi(n)$ может быть представлен произведением $n = pq$, где p и q – простые числа.

Таким образом, будет справедливо равенство:

$$\varphi(n) = (p-1)(q-1). \quad (47)$$

Тогда выражение (1) с учетом (47) может быть записано в виде:

$$a^{(p-1)(q-1)} \equiv 1 \pmod{pq}. \quad (48)$$

Параметр n будет выступать в роли модуля, который ограничивает размер информационного сообщения m , которое подвергается обработке, так что будет справедливо условие $1 < m < n$.

Открытый ключ E , называемый открытой экспонентой, выбирается таким образом, что удовлетворяет условию $1 < E < \varphi(n)$, при этом E и $\varphi(n)$ являются взаимно простыми.

Закрытый ключ D (закрытая экспонента), аналогично ключу E , также удовлетворяет условию $1 < D < \varphi(n)$ и является взаимно простым с $\varphi(n)$. Вместе с тем между ключами E и D существует связь, которую можно выразить следующим образом:

$$DE \equiv 1 \pmod{\varphi(n)}. \quad (49)$$

Из выражения (49) видно, что числа D и E являются обратными:

$$D \equiv E^{-1} \pmod{\varphi(n)}. \quad (50)$$

Для нахождения закрытой экспоненты D как вариант может быть использован расширенный алгоритм Евклида, приведенный в подразделе 1.2.3.

Процедура защиты информационного сообщения m строится на его последовательном возведении в степени E и D с приведением результатов по модулю n . Порядок использования ключей определяется задачей алгоритма: в случае с шифрованием – сначала E , затем D , для организации цифровой подписи – сначала D , затем E ,

В общем виде порядок действий следующий:

$$\begin{aligned} \text{шифрование:} \quad & X = m^E \pmod{n}; \quad m = X^D \pmod{n}; \\ \text{цифровая подпись:} \quad & Y = m^D \pmod{n}; \quad m = Y^E \pmod{n}. \end{aligned} \quad (51)$$

Приведем простейший пример, где $p = 3$ и $q = 11$. Сформируем для этого случая пару ключей.

Определим модуль:

$$n = pq = 3 \cdot 11 = 33.$$

Вычислим:

$$\varphi(n) = \varphi(33) = (p-1)(q-1) = 2 \cdot 10 = 20.$$

Выберем открытую экспоненту:

$$1 < E < 20: \quad E = 7.$$

Определим на основе (50) закрытую экспоненту:

$$D = 7^{-1} \equiv 3 \pmod{20}.$$

Проверим работу алгоритма. Для этого выберем некоторое произвольное число $m = 5$, удовлетворяющее условию $1 < m < n$.

Произведем в соответствии с (51) операцию возведения m в степень E с приведением результата по $\text{mod } n$:

$$X = m^E = 5^7 = 78125 \equiv 14 \pmod{33}.$$

Теперь в соответствии с (51) произведем восстановление m путем возведения X в степень D с приведением результата по $\text{mod } n$:

$$M = X^D = 14^3 = 2744 \equiv 5 \pmod{33}.$$

Равенство $M = m = 5$ подтверждает правильную работу алгоритма.

Аналогичная ситуация наблюдается и в случае обратного применения ключей. Сначала в соответствии с (51) возведем m в степень D и приведем результат по $\text{mod } n$:

$$Y = m^D = 5^3 = 125 \equiv 26 \pmod{33}.$$

Произведем в соответствии с (51) восстановление m путем возведения Y в степень E и приведем результат по $\text{mod } n$:

$$M = Y^E = 26^7 = 8031810176 \equiv 5 \pmod{33}.$$

Снова убедимся в правильной работе алгоритма.

3.5.2. Цифровая подпись на основе RSA

Цифровая подпись в криптосистеме RSA обычно применяется в соответствии с основным принципом алгоритма DSA, где подписывается не само информационное сообщение, а только его хеш (рис. 27).

Пусть стороне A необходимо подтвердить свою подлинность при отправке стороне B некоторого информационного сообщения m .

Тогда на стороне A создается пара ключей: E и D на основе двух больших простых чисел: p и q . Открытая экспонента E вместе с модулем n публикуется и таким образом становится доступной стороне B .

Сторона A подписывает сообщение своим закрытым ключом D . Сформированная цифровая подпись Y вместе с информационным сообщением m передается стороне B по открытому каналу связи.

Сторона B производит обработку цифровой подписи Y , вычисляя предполагаемое значение информационного сообщения M .

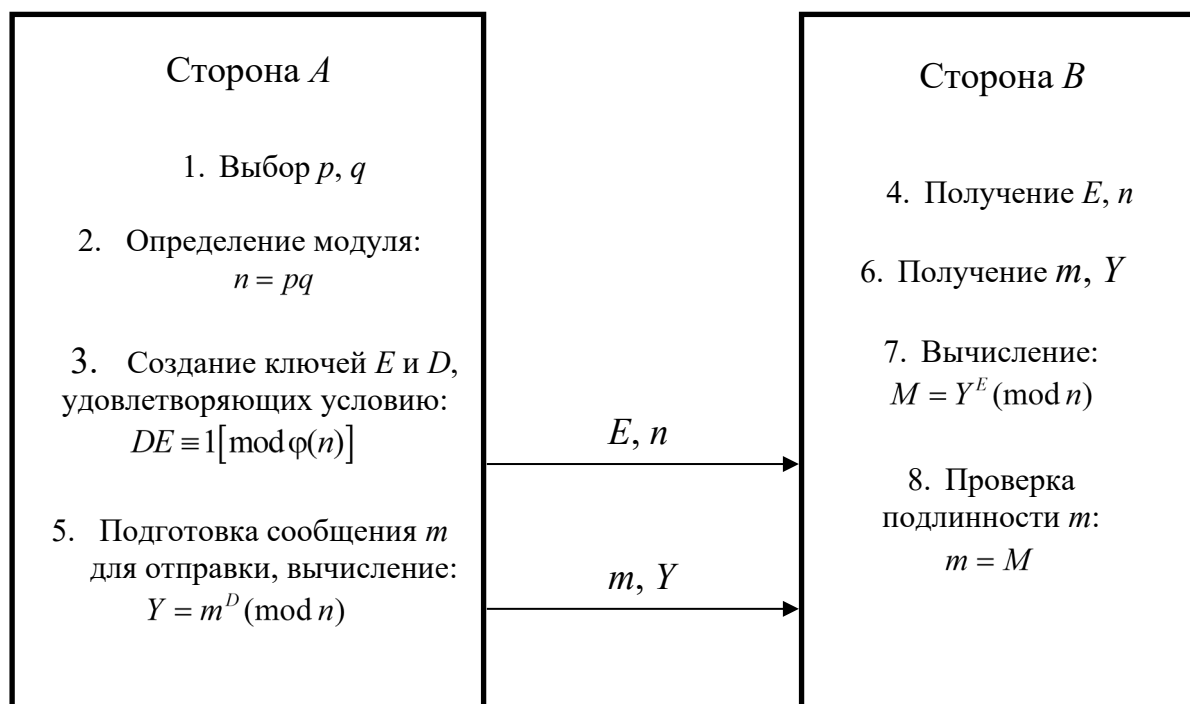


Рис. 27. Алгоритм цифровой подписи в RSA

Сравнивая принятое сообщение m с результатом обработки его цифровой подписи M , сторона B принимает решение о том, было ли данное сообщение подписано стороной A .

3.5.3. Шифрование на основе RSA

Процедура шифрования в криптосистеме RSA может быть построена по тем же принципам, что и в схеме Эль-Гамала (рис. 28).

Пусть сторона A должна передать стороне B некоторое зашифрованное сообщение m , которое удовлетворяет условию $1 < m < n$. Поэтому именно сторона B инициирует создание ключей и является единственным обладателем закрытой экспоненты D , которую формирует сама или получает из доверительного источника по безопасному каналу связи.

Сторона A производит шифрование сообщения m при помощи открытой экспоненты E . Зашифрованное сообщение X передается стороне B по открытому каналу связи.

Сторона B принимает сообщение X , производит его обработку при помощи закрытой экспоненты D и таким образом получает информационное сообщение m .

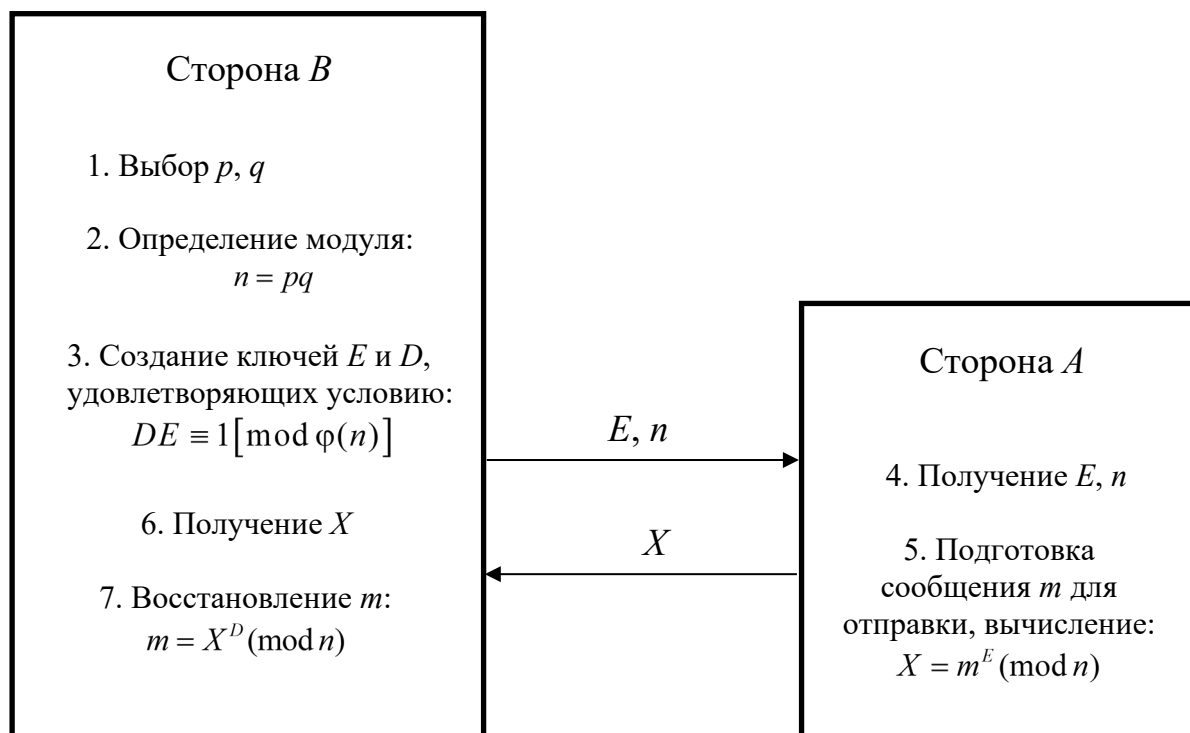


Рис. 28. Алгоритм цифровой подписи в RSA

Стоит отметить, что на практике данная схема шифрования данных не используется, так как результат возведения одного и того же информационного сообщения m в степень E будет одинаковым. Связано это с тем, что экспоненты E и D являются постоянными. Такой шифр будет обладать низкой надежностью.

Поэтому алгоритм шифрования RSA используется в современных системах передачи данных, как правило, для безопасной передачи общего секретного ключа, который в рамках отдельно взятого сеанса связи будет обеспечивать работу своего симметричного алгоритма шифрования.

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

- 1) самостоятельно сформировать первый ключ на основе параметров p и q из табл. 8;
- 2) при помощи расширенного алгоритма Евклида определить второй ключ;
- 3) зашифровать и расшифровать сообщение m из табл. 8, используя пару самостоятельно созданных ранее ключей.

Таблица 8

Варианты заданий согласно номеру в журнале обучающихся

№ варианта	p	q	m	№ варианта	p	q	m
1	11	37	54	16	71	29	8
2	19	29	35	17	23	61	9
3	41	17	24	18	17	73	15
4	109	7	31	19	181	7	18
5	13	127	9	20	137	19	4
6	101	71	3	21	157	5	36
7	53	59	4	22	107	43	5
8	47	11	12	23	79	19	14
9	5	179	15	24	7	199	12
10	307	3	11	25	193	17	7
11	61	79	2	26	53	61	6
12	127	17	6	27	269	5	15
13	19	29	19	28	149	23	4
14	73	23	14	29	79	83	3
15	53	41	25	30	37	109	6

3.6. Схема ECDSA

Надежность криптосистемы ECDSA (Elliptic Curve Digital Signature Algorithm – алгоритм проверки цифровой подписи на эллиптической кривой) обеспечивается сложностью вычисления дискретного логарифма в конечном поле, элементами которого являются точки эллиптической кривой. Таким образом, имея две точки P и Q , принадлежащие некоторой эллиптической кривой, достаточно сложно найти такое число k , при котором выполняется равенство $Q = kP$.

Алгоритм ECDSA, используемый при построении блокчейна, задает эллиптическую кривую над конечным полем и некоторую базовую точку. Например, в Биткойне используется кривая secp256k1 с параметрами [27]:

- порядок конечного поля: $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$;
- параметры эллиптической кривой: $a = 0$, $b = 7$;
- базовая точка $G =$

04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:0b:07:02:9b:fc:db:2d:ce:28:d9:59:
f2:81:5b:16:f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:0e:11:08:a8:fd:17:b4:
48:a6:85:54:19:9c:47:d0:8f:fb:10:d4:b8;

- для точки G порядок подгруппы $n =$
00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:
36:41:41;
- кофактор $h = 1$.

Таким образом, для создания ключей в алгоритме ECDSA используется набор следующих параметров: p , a , b , G , n , h .

Закрытый ключ является случайным целым числом d , удовлетворяющим условию:

$$1 \leq d \leq n - 1.$$

Открытый ключ связан с закрытым ключом через базовую точку:

$$H = dG. \quad (52)$$

Таким образом, в ECDSA закрытым ключом является d , в качестве открытого ключа выступает точка H . Это правило особенно распространяется на системы, использующие блокчейн, где эллиптическая кривая заранее определена, поэтому открытым ключом здесь действительно будет точка H , полученная, как показано выше, на основе равенства (52).

Однако на практике алгоритм цифровой подписи для большей надежности может использовать случайную эллиптическую кривую. В этом случае открытым ключом уже будет целый набор параметров (p, a, b, G, n, H) , закрытым ключом останется число d .

3.6.1. Цифровая подпись на основе схемы ECDSA

Основной задачей здесь, как и в аналогичных схемах цифровой подписи, является аутентификация отправителя сообщения с сохранением целостности передаваемых данных. Можно считать данную схему некоторой модификацией алгоритма DSA.

Согласно алгоритму, сторона A , подтверждающая свою подлинность, формирует два ключа: закрытый ключ d и на его основе открытый ключ H (рис. 29).

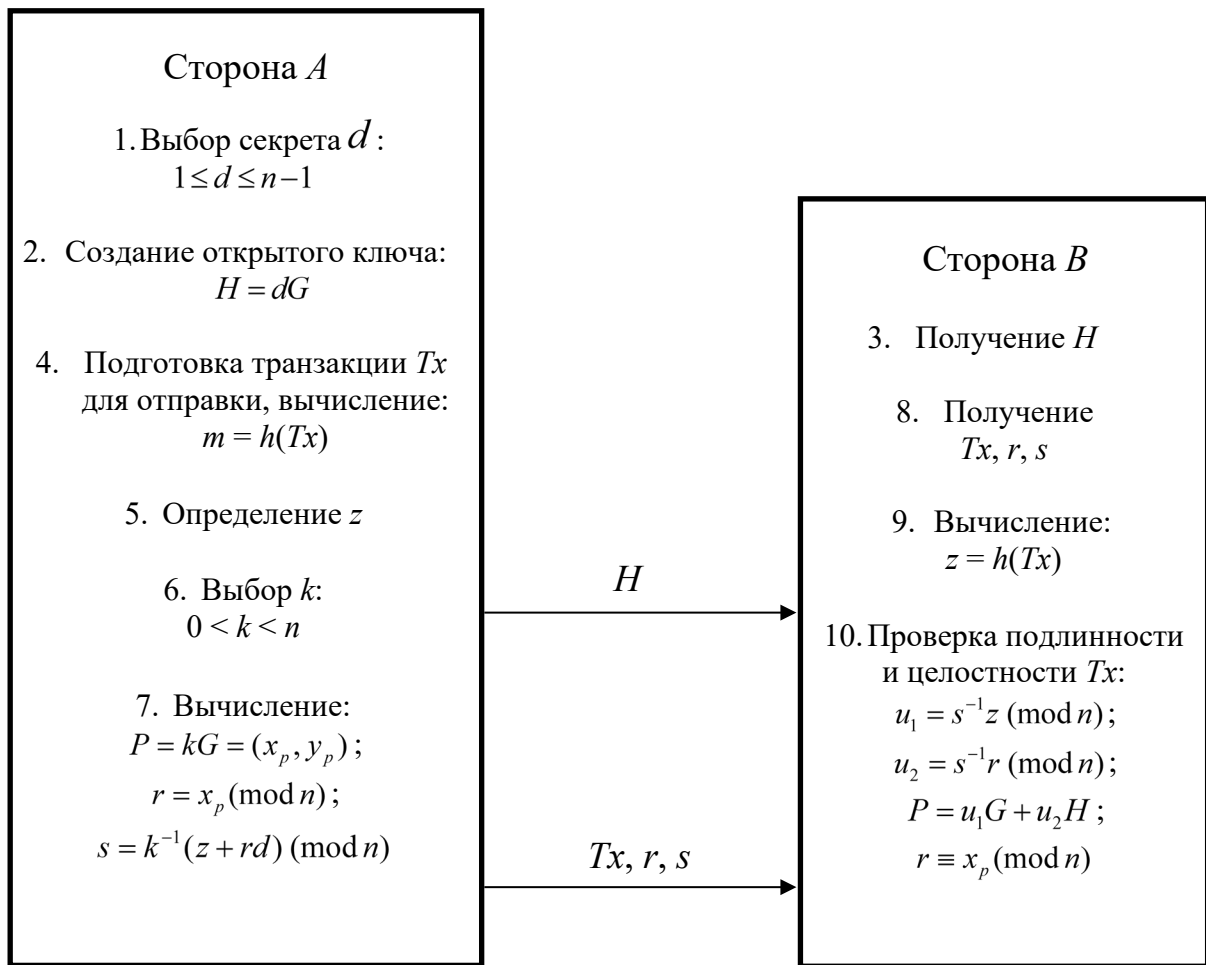


Рис. 29. Алгоритм цифровой подписи на основе схемы ECDSA

Ключ H в соответствии со схемой (рис. 29) должен быть общедоступным. Таким образом, H публикуется или передается стороне B по открытому каналу связи.

Пусть сторона A планирует отправку некоторого сообщения стороне B , например, транзакции Tx в рамках блокчейна, использующего схему ECDSA. При этом стороне A требуется решить две задачи: защитить Tx от подмены и доказать стороне B свое авторство по отношению к данной транзакции.

Защита подлинности транзакции Tx эффективно решается при помощи некоторой функции хеширования $h(Tx)$. Таким образом, формируется дайджест сообщения Tx : $m = h(Tx)$.

В соответствии с принципом DSA сторона A должна подписать свою транзакцию с помощью закрытого ключа d . Сторона B проверит подпись стороны A с помощью открытого ключа H и вместе с тем убедится в том, что транзакция избежала подмены.

Таким образом, алгоритм защиты транзакции будет следующим.

1. Вычисляется хеш транзакции $m = h(Tx)$.
2. Хеш m нормируется таким образом, чтобы его битовая длина не превышала порядок подгруппы n . Полученный урезанный хеш будет обозначаться некоторым параметром z .
3. Выбирается случайное целое число k , удовлетворяющее условию $0 < k < n$.

4. Вычисляется точка:

$$P = kG = (x_p, y_p). \quad (53)$$

5. Вычисляется:

$$r = x_p \pmod{n}. \quad (54)$$

6. Если $r = 0$, то происходит возвращение к шагу 3.

7. Вычисляется параметр:

$$s = k^{-1}(z + rd) \pmod{n}. \quad (55)$$

8. Если $s = 0$, то происходит возвращение к шагу 3.

Полученная в результате вычислений пара (r, s) выступает в роли цифровой подписи.

Получив транзакцию Tx вместе с цифровой подписью (r, s) , сторона B должна удостовериться в том, что автором данной транзакции действительно является сторона A , и сама транзакция не была подменена при передаче по открытому каналу связи.

Сторона B верифицирует транзакцию, используя следующий алгоритм.

1. На основе принятого значения Tx вычисляется хеш $m = h(Tx)$.
2. Хеш m нормируется до урезанного значения z , чтобы его битовая длина не превышала n .
3. Вычисляется параметр:

$$u_1 = s^{-1}z \pmod{n}. \quad (56)$$

4. Вычисляется параметр:

$$u_2 = s^{-1}r \pmod{n}. \quad (57)$$

5. Определяется точка:

$$P = u_1G + u_2H. \quad (58)$$

6. Подпись является действительной, если выполняется условие:

$$r \equiv x_p \pmod{n}.$$

Следует заметить, что число k , выбранное стороной A , играет важную роль в вопросе надежности цифровой подписи. Желательно, чтобы каждая

новая процедура подписи хеша использовала новое значение параметра k , которое при этом должно тщательно храниться на стороне A . Использование одного и того же значения k для двух разных подписей приводит к его простому вычислению.

Допустим, имеются хеши z_1 и z_2 двух различных транзакций. Для них на основе выражений (53)–(55) вычислены подписи соответственно (r_1, s_1) и (r_2, s_2) .

Предположим, что данные подписи были получены с использованием одного числа k . Таким образом, в силу (53) и (54) будет справедливо равенство $r_1 = r_2$. Из этого следует равенство:

$$(s_1 - s_2) = k^{-1}(z_1 - z_2) \pmod{n}. \quad (59)$$

Умножив обе части (59) на параметр k и разделив на $(s_1 - s_2)$, получим:

$$k(s_1 - s_2) = (z_1 - z_2) \pmod{n}. \quad (60)$$

Выражение (60) позволяет легко вычислить значение параметра k :

$$k = (z_1 - z_2)(s_1 - s_2)^{-1} \pmod{n}.$$

Рассмотрим работу алгоритма ECDSA на примере следующей эллиптической кривой:

$$y^2 = x^3 + 2x + 3 \pmod{17},$$

которая имеет следующие параметры: $a = 2, b = 3, p = 17$.

Определим основные параметры для формирования ключей:

- базовая точка $G = (3, 6)$;
- порядок подгруппы базовой точки $n = 11$;
- кофактор $h = 2$.

В качестве закрытого ключа выберем случайное число от 1 до $n - 1$, например, $d = 3$. Тогда открытый ключ в соответствии с правилом (52) будет представлять собой точку: $H = dG = 3G = (15, 5)$.

Пусть подготовленный для подписи хеш транзакции имеет значение $z = 5$. Тогда процедура подписи транзакции стороной A будет следующей.

Определяется случайное $k = 7$, которое удовлетворяет условию $0 < k < n$.

1. На основании (53) вычисляется: $P = kG = 7G = (14, 15)$.

2. На основании (54) вычисляется: $r = x_p = 14 = 3 \pmod{11}$.

3. На основании (55) вычисляется:

$$s = k^{-1}(z + rd) = 7^{-1}(5 + 3 \cdot 3) = 8 \cdot 14 = 112 = 2 \pmod{11}.$$

Подпись хеша транзакции будет иметь вид: $(r, s) = (3, 2)$.

Сторона B , обладая открытым ключом $H = (15, 5)$, получив транзакцию и (r, s) , начинает проверку подписи:

- 1) вычисляется хеш транзакции $z = 5$;
- 2) на основе (56) вычисляется $u_1 = s^{-1}z = 2^{-1} \cdot 5 = 6 \cdot 5 = 30 = 8 \pmod{11}$;
- 3) на основе (57) вычисляется $u_2 = s^{-1}r = 2^{-1} \cdot 3 = 6 \cdot 3 = 18 = 7 \pmod{11}$;
- 4) на основе (58) вычисляется

$$P = u_1G + u_2H = 8G + 7H = (15, 12) + (3, 11) = (14, 15).$$

Теперь не составляет труда убедиться в справедливости тождества:

$$r \equiv x_p \pmod{11}, \quad r = 3, x_p = 3,$$

которое подчеркивает действительность подписи.

3.6.2. Поддержка ECDSA в OpenSSL

В библиотеке *OpenSSL* представлен большой список рекомендованных эллиптических кривых для шифрования и цифровой подписи. Данный список отображается командой **openssl ecparam -list_curves**:

```
$ openssl ecparam -list_curves

secp112r1 : SECG/WTLS curve over a 112 bit prime field
secp112r2 : SECG curve over a 112 bit prime field
secp128r1 : SECG curve over a 128 bit prime field
secp128r2 : SECG curve over a 128 bit prime field
secp160k1 : SECG curve over a 160 bit prime field
secp160r1 : SECG curve over a 160 bit prime field
secp160r2 : SECG/WTLS curve over a 160 bit prime field
secp192k1 : SECG curve over a 192 bit prime field
secp224k1 : SECG curve over a 224 bit prime field
secp224r1 : NIST/SECG curve over a 224 bit prime field
secp256k1 : SECG curve over a 256 bit prime field
secp384r1 : NIST/SECG curve over a 384 bit prime field
secp521r1 : NIST/SECG curve over a 521 bit prime field
prime192v1: NIST/X9.62/SECG curve over a 192 bit prime field
prime192v2: X9.62 curve over a 192 bit prime field
prime192v3: X9.62 curve over a 192 bit prime field
prime239v1: X9.62 curve over a 239 bit prime field
.....
brainpoolP160t1: RFC 5639 curve over a 160 bit prime field
brainpoolP192r1: RFC 5639 curve over a 192 bit prime field
brainpoolP192t1: RFC 5639 curve over a 192 bit prime field
brainpoolP224r1: RFC 5639 curve over a 224 bit prime field
```

brainpoolP224t1: RFC 5639 curve over a 224 bit prime field
 brainpoolP256r1: RFC 5639 curve over a 256 bit prime field
 brainpoolP256t1: RFC 5639 curve over a 256 bit prime field
 brainpoolP320r1: RFC 5639 curve over a 320 bit prime field
 brainpoolP320t1: RFC 5639 curve over a 320 bit prime field
 brainpoolP384r1: RFC 5639 curve over a 384 bit prime field
 brainpoolP384t1: RFC 5639 curve over a 384 bit prime field
 brainpoolP512r1: RFC 5639 curve over a 512 bit prime field
 brainpoolP512t1: RFC 5639 curve over a 512 bit prime field
 SM2 : SM2 curve over a 256 bit prime field

Список содержит более 80 названий кривых с их кратким описанием. Эллиптическая кривая `secp256k1`, рассмотренная выше, также присутствует в данном списке.

Параметры отдельно взятой кривой выводятся при помощи команды: **`openssl ecparam -name [имя кривой] -noout -text -param_enc explicit`**. Добавление ключа **`-noout`** позволяет не выводить параметры в закодированном виде, а ключ **`-text`** выводит их в текстовой форме.

Для примера выведем на экран параметры кривой `secp256k1`:

```
$ openssl ecparam -name secp256k1 -noout -text -param_enc explicit
```

Field Type: prime-field

Prime:

00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:

ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ff:

ff:fc:2f

A: 0

B: 7 (0x7)

Generator (uncompressed):

04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:

0b:07:02:9b:fc:db:2d:ce:28:d9:59:f2:81:5b:16:

f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:

0e:11:08:a8:fd:17:b4:48:a6:85:54:19:9c:47:d0:

8f:fb:10:d4:b8

Order:

00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:

ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:

36:41:41

Cofactor: 1 (0x1)

В результате станет доступной следующая информация:

- Prime – порядок конечного поля;
- A, B – параметры уравнения кривой;

- Generator (uncompressed) – базовая точка в несжатом виде;
- Order – порядок подгруппы базовой точки;
- Cofactor – кофактор.

Иногда базовую точку можно представить в «сжатом» виде, указав только x координату, так как необязательная y координата всегда может быть вычислена через уравнение:

$$y^2 = x^3 + ax + b \pmod{p}.$$

Процедура генерации закрытого ключа осуществляется командой **openssl ecparam -name [имя кривой] -genkey -noout -out [имя файла]**, где параметр **-out** задает путь к файлу, в который будет помещен закрытый ключ:

```
$ openssl ecparam -name secp256k1 -genkey -noout -out private-
key.pem
$ cat privatekey.pem

-----BEGIN EC PRIVATE KEY-----
MHQCAQEEIIIVduxp2MDoZww9MGMLUppE+Foh88RjGyrIPLIyiTe2oAcGBSuBBAK
oUQDQgAEmR1zfZMCo1bdyZhCkUNe20qRipwX42xXsS0cRvrFfsm/seGbz65iPthI
rSt2SJBjPPh13efKmkSNG2/KJwmqIA==
-----END EC PRIVATE KEY-----
```

Генерация открытого ключа на основе закрытого предполагает использование команды вида **openssl ec -in [имя файла с закрытым ключом] -pubout -out [имя файла для открытого ключа]**:

```
$ openssl ec -in privatekey.pem -pubout -out publickey.pem
$ cat publickey.pem

-----BEGIN PUBLIC KEY-----
MFYwEAYHKoZIzj0CAQYFK4EE-
AAoDQgAEmR1zfZMCo1bdyZhCkUNe20qRipwX42xX
sS0cRvrFfsm/seGbz65iPthIrSt2SJBjPPh13efKmkSNG2/KJwmqIA==
-----END PUBLIC KEY-----
```

Закрытый ключ позволяет подписать целевой файл по схеме ECDSA. Для примера организуем процедуру цифровой подписи для некоторого произвольного файла. Пусть это будет файл *Tx.txt*, содержащий строку: «test_tx».

Подпись файла производится командой следующего вида:

openssl dgst -[имя хеш алгоритма] -sign [путь к файлу с закрытым ключом] [файл для подписи] > [имя файла для сохранения подписи].

Алгоритм ECDSA предполагает подпись не самого текста, содержащегося в файле, а его хеша, поэтому в параметрах команды задается имя алгоритма, который будет хешировать содержимое файла перед подписью. После параметра **[файла для подписи]** следует символ «>», который необходим для того, чтобы сохранить полученную подпись в новый файл.

Подпишем хеш (алгоритм SHA256) файла *Tx.txt*:

```
$ openssl dgst -sha256 -sign privatekey.pem Tx.txt > sign.bin
```

Подпись была сохранена в файле *sign.bin*, содержимое которого будет выглядеть, как показано на рис. 30.

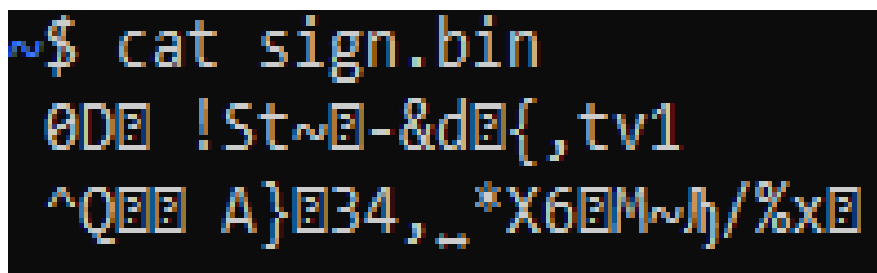


Рис. 30. Содержимое файла с подписью

Обладая открытым ключом и подписью, не составляет труда верифицировать данные с помощью команды следующего вида:

openssl dgst -[имя хеш алгоритма] -verify [путь к файлу с открытым ключом] -signature [путь к файлу с подписью] [путь к файлу с данными].

Очевидно, что параметр **[имя хеш алгоритма]** должно совпадать с тем, которое было использовано для хеширования в процессе выполнения процедуры подписания файла.

Верифицируем подписанный файл *Tx.txt*:

```
$ openssl dgst -sha256 -verify publickey.pem -signature  
sign.bin Tx.txt
```

```
Verified OK
```

Если перед проверкой подписи внести преднамеренные изменения в содержимое подписанного файла, то верификация будет отрицательной:

```
$ echo -n new_data > Tx.txt  
$ openssl dgst -sha256 -verify publickey.pem -signature  
sign.bin Tx.txt
```

```
Verification Failure
```

Верификация также будет безуспешной в случае использования другого публичного ключа:

```
$ openssl ecparam -name secp256k1 -genkey -noout -out private-key-2.pem
$ openssl ec -in privatekey-2.pem -pubout -out publickey-2.pem
$ openssl dgst -sha256 -verify publickey-2.pem -signature sign.bin Tx.txt
```

Verification Failure

Задание на самостоятельную работу

Настоятельно рекомендуется индивидуальное выполнение данного задания. По результатам должен быть сформирован соответствующий отчет в рукописном, печатном или электронном виде.

В качестве задания требуется:

- 1) выбрать из табл. 9 исходные данные, где p – порядок конечного поля, a, b – параметры уравнения кривой, G – базовая точка, N – порядок группы эллиптической кривой;
 - 2) записать уравнение эллиптической кривой по данным параметрам;
 - 3) найти порядок подгруппы, генерируемой точкой G . Найти кофактор полученной подгруппы;
 - 4) самостоятельно сгенерировать закрытый и открытый ключи на основе параметров эллиптической кривой;
 - 5) на основе полученного закрытого ключа подписать урезанный хеш транзакции z ;
 - 6) верифицировать полученную подпись;
 - 7) зафиксировать параметры стандартизированной кривой из табл. 5, используя *OpenSSL*;
 - 8) сгенерировать закрытый и открытый ключи, используя данную стандартизированную кривую в *OpenSSL*;
 - 9) создать текстовый файл со строками, содержащими информацию об учащемся (Ф. И. О., название группы, номер варианта);
 - 10) подписать по схеме ECDSA данный файл, используя сгенерированный закрытый ключ. Использовать алгоритм хеширования из табл. 9;
 - 11) верифицировать полученную подпись.
- Изменить содержимое исходного текстового файла и произвести верификацию подписи снова.

Таблица 9

Варианты заданий согласно номеру в журнале обучающихся

№ вар.	Параметры эллиптической кривой для ручного расчета					Урезанный хеш транзакции z	Имя кривой	Алгоритм хеширования
	p	a	b	G	N			
0	89	4	1	(77, 19)	87	19	secp112r1	md5
1	31	6	5	(5, 6)	30	71	prime192v2	sha224
2	59	3	8	(43, 17)	50	53	brainpoolP256r1	sha512-256
3	61	5	0	(58, 18)	72	100	secp224k1	sha3-256
4	23	-2	0	(14, 18)	24	79	prime239v1	rmd160
5	97	3	-3	(13, 14)	96	20	brainpoolP512t1	sha256
6	89	6	-5	(1, 25)	90	78	SM2	sha3-512
7	37	10	2	(23, 35)	40	58	secp384r1	md5-sha1
8	41	10	-1	(4, 12)	48	84	prime256v1	sha1
9	43	0	-7	(0, 6)	57	42	brainpoolP320r1	whirlpool

КОНТРОЛЬНЫЕ ВОПРОСЫ

Раздел 1

1. Перечислите основные направления по защите информации.
2. В чем состоит разница между симметричным и асимметричным шифрованием?
3. Что определяет функция Эйлера?
4. Какие элементы мультипликативной группы кольца вычетов по заданному модулю образуются на основе первообразного элемента?
5. В чем состоит физический смысл малой теоремы Ферма?
6. Какими способами может быть определен обратный элемент мультипликативной группы?
7. Какие криптосистемы используют дискретный логарифм?
8. Какие математические операции применяются в полях Галуа?
9. Какое математическое условие является необходимым для того, чтобы кривая на плоскости являлась эллиптической кривой?
10. Какие математические операции характерны для точек эллиптической кривой?
11. Как связаны эллиптические кривые с конечными полями?
12. Какая криптосистема использует эллиптические кривые?

Раздел 2

1. Какие задачи решают функции хеширования?
2. Каков основной механизм построения функций хеширования?
3. Перечислите требования, которые предъявляются к современным функциям хеширования.
4. Какими основными свойствами обладают функции хеширования?
5. Какие группы функций хеширования известны на сегодняшний день?
6. Что представляет собой OpenSSL?
7. Какова роль функций хеширования в сетях блокчейн?
8. Для решения каких задач используется сеть блокчейн?
9. Изобразите простейшую структуру блокчейн.
10. В чем заключается механизм поддержания безопасности транзакции в сети блокчейн?
11. Какие криптосистемы используются в современных сетях блокчейн?
12. Какую роль играет дерево Меркла в работе с сетью блокчейн?

Раздел 3

1. Что является симметричным ключом в шифре Виженера?
2. Какие современные сетевые технологии обеспечивают безопасность при помощи криптосистемы Диффи–Хеллмана?
3. На чем основана безопасность криптосистемы Диффи–Хеллмана?
4. Какова область применения схемы разделения секрета Шамира?
5. На чем основана безопасность криптосистемы, которая обеспечивает разделение секрета Шамира?
6. Какие задачи информационной безопасности решает криптосистема Эль-Гамала?
7. Какие современные сетевые технологии обеспечивают безопасность при помощи криптосистемы RSA?
8. Какие задачи информационной безопасности решает криптосистема RSA?
9. На чем основана безопасность криптосистемы RSA?
10. Какие современные сетевые технологии обеспечивают безопасность с помощью криптосистемы ECDSA?
11. Какие задачи информационной безопасности решает криптосистема ECDSA?
12. На чем основана безопасность криптосистемы ECDSA?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Когновицкий, О. С. Двойственный базис и его применение в телекоммуникациях / О. С. Когновицкий. – Санкт-Петербург : Линк, 2009.
2. Кукунин, Д. С. Перспективы использования рекуррентных последовательностей в современной телекоммуникационной среде : [монография] / Д. С. Кукунин, О. С. Когновицкий, А. А. Березкин, Р. В. Киричек ; СПбГУТ. – Санкт-Петербург, 2023. – 289 с.
3. Лидл, Р. Конечные поля : В 2-х т. Т. 1 : пер. с англ. / Р. Лидл, Г. Нидеррайтер. – Москва : Мир, 1988. – 430 с.
4. Coppersmith, D. Discrete logarithms in $GF(p)$ / D. Coppersmith, A. M. Odlyzko, R. Schroepel // *Algorithmica*. – 1986. – Vol. 1. – № 1. – P. 1–15.
5. Gordon, D. M. Discrete logarithms in $GF(p)$, using the number field sieve / D. M. Gordon // *SIAM J. Disc. Math.* – 1993. – Vol. 6. – № 1. – P. 124–138.
6. McCurley, K. S. The discrete logarithm problem / K. S. McCurley // *Proc. of Symp. in Appl. Math.* – 1990. – Vol. 42. – P. 49–74.
7. Mironov, I. Hard Instances of the Constrained Discrete Logarithm Problem / I. Mironov, A. Mityagin, K. Nissim // 7th Algorithmic Number Theory Symposium (ANTS VII), LNCS 4076. – 2006. – P. 582–598.
8. Granger, R. On the Discrete Logarithm Problem on Algebraic Tori / R. Granger, F. Vercauteren // *Crypto 2005*, LNCS 3621. – 2005. – P. 66–85.
9. Когновицкий, О. С. Метод вычисления логарифма в конечном поле / О. С. Когновицкий, В. Н. Сюрин, Б. А. Асанович // IX Всесоюзная конференция по теории кодирования и передачи информации : Тезисы докладов. Часть 1. – Одесса, 1988.
10. Кукунин, Д. С. Дискретное логарифмирование в полях Галуа с использованием контрольных точек / Д. С. Кукунин // Научно-технические ведомости / СПбГПУ. – Санкт-Петербург, 2009. – № 2. – С. 186–192.
11. Coppersmith, D. Fast evaluation of logarithms in fields of characteristic two / D. Coppersmith // *IEEE Transaction on Information Theory*. – 1984. – Vol. 30. – P. 587–594.
12. LaMacchia, B. A. Computation of discrete logarithms in prime fields / B. A. LaMacchia, A. M. Odlyzko // *Designs, Codes, and Cryptography*. – 1991. – № 1. – P. 46–62.
13. Adleman, L. M. A subexponential algorithm for discrete logarithms over all finite fields / L. M. Adleman, J. DeMarrais // *Mathematics of Computation*. – 1993. – № 203. – P. 1–15.
14. Имран, Б. Блокчейн: архитектура, криптовалюта, инструменты разработки, смарт-контракты [Текст] / Б. Имран. – Москва : ДМК Пресс, 2019. – 538 с.
15. Острик, В. В. Алгебраическая геометрия и теория чисел: рациональные и эллиптические кривые / В. В. Острик, М. А. Цфасман. – Москва : МЦНМО, 2001. – 48 с.
16. Коблиц, Н. Курс теории чисел и криптографии / Н. Коблиц. – Москва : Научное издательство «ТПВ», 2001. – 254 с.
17. Бабенко, М. Г. Анализ методов скалярного умножения на эллиптической кривой / М. Г. Бабенко // Молодой ученый. – 2010. – с. 24–29.
18. Зорич, В. А. Математический анализ. Часть 1 / В. А. Зорич. – 6-е издание, дополненное. – Москва : МЦНМО, 2012. – 711 с.
19. Болотов, А. А. Элементарное введение в эллиптическую криптографию : Алгебраические и алгоритмические основы / А. А. Болотов, С. Б. Гашков, А. Б. Фролов, А. А. Часовских. – Москва : КомКнига, 2006. – Т. 1. – 328 с.
20. FIPS 186-4 – Gaithersburg: NIST, 2013. – 130 с.

21. *Martin, Keith M.* Everyday Cryptography / Keith M. Martin. – Oxford University Press, 2012. – P. 142.
22. *Diffie, W.* New Directions in Cryptography / W. Diffie, M. E. Hellman // IEEE Transactions on Information Theory. – 1976. – Vol. IT-22, № 6. – P. 644–654.
23. *Shamir, A.* How to share a secret / A. Shamir // Communications of the ACM – New York City: Association for Computing Machinery, 1979. – Vol. 22, Iss. 11. – P. 612–613.
24. *Авхадиев, Ф. Г.* Учебно-методическое пособие по численным методам анализа / Ф. Г. Авхадиев, Р. К. Губайдуллина, Р. Г. Насибуллин. – Казань : Казанский (Приволжский) федеральный университет, 2019. – 113 с.
25. *ElGamal, T.* A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms / T. ElGamal // IEEE Transactions on Information Theory. – 1985. – Vol. IT-31, № 4. – P. 469–472.
26. *Rivest R., Shamir A., Adleman L.* A method for obtaining digital signatures and public-key cryptosystems (англ.) // *Communications of the ACM* – New York City: Association for Computing Machinery, 1978. – Vol. 21, Iss. 2. – P. 120–126.
27. Elliptic Curve Digital Signature Algorithm [Электронный ресурс] : https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm.