КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра системного анализа и информационных технологий

Ш.Т. ИШМУХАМЕТОВ, Р.Х. ЛАТЫПОВ,
Р.Г. РУБЦОВА, Е.Л. СТОЛОВ

ВВЕДЕНИЕ В ТЕОРИЮ КОДИРОВАНИЯ И КРИПТОГРАФИЮ

Учебное пособие

Казань 2021

УДК 512.624
ББК 22.144

Принято на заседании кафедры системного анализа и информационных технологий
Протокол № 5 от 10 февраля 2021

Принято на заседании учебно-методической комиссии Института вычислительной математики и информационных технологий
Протокол № 8 от 19 марта 2021

РЕЦЕНЗЕНТЫ:
доктор физико-математических наук, профессор, зав.каф. теоретической кибернетики **Ф.М. Аблаев**
кандидат физико-математических наук, доцент каф. теоретической кибернетики **В.С. Кугураков**

Ишмухаметов Ш.Т., Латыпов Р.Х., Рубцова Р.Г., Столов Е.Л. Введение в теорию кодирования и криптографию: Учебное пособие/ Ш.Т. Ишмухаметов, Р.Х. Латыпов, Р.Г. Рубцова Е.Л. Столов, . – Казань: Казанский ун-т, 2021.- 211 с.

В учебном издании представлены материалы для изучения базовых понятий, связанных с конечными полями, кодированием информации, выполнен разбор примеров, приведены приложения к криптографии и теории чисел. Учебник может быть использован для самостоятельной подготовки по курсу "Математические основы информационной безопасности".

Оглавление

ВВЕДЕНИЕ

Криптография - это наука, занимающаяся построением безопасных шифров, т.е. алгоритмов, обеспечивающих преобразование электронных документов в нечитаемый набор символов, из которого можно восстановить исходный документ только зная некоторый пароль (секретное слово). Криптография развивалась с самого начала истории человечества. Известны примеры шифрования текстов из глубокой древности - у римлян, греков и других народов.

Однако эра современной криптографии началась сравнительно недавно, в 70-е годы XX столетия. Приведем здесь основные события тех лет.

В 1977 г. трое ученых Рональд Райвест (Ronald Linn Rivest), Ади Шамир (Adi Shamir) и Леонард Адлеман (Leonard Adleman) из Массачусетского Технологического Института (MIT) опубликовали в журнале Scientific American новый алгоритм шифрования, основанный на идее двухключевого шифрования, названный по первым буквам фамилий авторов методом RSA. В этом методе известным параметром служит некоторое целое число n большой длины (обычно 1024 или 2048 бита), являющееся произведением двух простых чисел p и q. Эти числа p и q являлись секретными параметрами метода, и для взлома системы RSA было достаточно найти множители p и q, т.е выполнить разложение числа n на простые сомножители.

На момент опубликования алгоритма RSA было известны лишь небольшое количество алгоритмов факторизации, самым известным из которых являлся метод Ферма. Эти методы позволяли на тот день факторизовать числа, состоящие не более чем из 25 – 30 цифр. Поэтому использование в качестве n натурального числа, имеющего более 100 десятичных знаков, гарантированно обеспечивало безопасность шифрования этим методом. Сами создатели метода предложили всей математической общественности для тестового взлома 129-значное десятичное число, пообещав за его разложение условное вознаграждение в $100. Масла в огонь подлила также опубликованная в 1977 г. в журнале Sci.Amer. статья известного математика и популяризатора Мартина Гарднера «A new kind of cipher that would take millions of years to break» («Новый алгоритм шифрования, для взлома которого потребуется миллионы лет») [18].

Однако через 17 лет 129-значное число создателей метода RSA было разложено на составные множители с помощью алгоритма квадратичного решета, реализованного в сети коллективом авторов, возглавляемым

А.Ленстрой. Эта процедура потребовала колоссальных усилий. Была задействована сеть, состоящая из 1600 компьютеров, которые проработав 220 дней, подготовили систему линейных уравнений, содержащую более 0,5 млн неизвестных. Потом эта система была решена с помощью суперкомпьютера за 2 дня вычислений.

Параллельно с методом RSA американцами У.Диффи и М. Хеллманом в 1976 году был разработан алгоритм, позволяющий вырабатывать общий секретный ключ для двух пользователей сети, общающихся через открытую сеть. Этот метод основывался на трудности задачи вычисления дискретного логарифма в конечных полях.

Схема построения электронной цифровой подписи на электронные документы, обеспечивающей те же условия, что и обычная собственноручная подпись, была разработана в 1984 году американстким криптографом Эль-Гамалем, и была развитием метода Диффи-Хеллмана.

Современное использование электронной подписи включает целую сферу услуг в бизнесе, экономике, промышленности. Современный документооборот не может обойтись без подтверждения личности подписывающего лица через глобальные сети, включая Интернет, которое может выполняться в автоматическом режиме с использованием криптографических ключей, выдаваемых сетью удостоверяющих центров.

Следующий серьезный этап в развитии криптографии связан с разработкой понятия *эллиптических кривых*. В 1985 В.Миллер и Н.Коблиц показали, что проблема вычисления кратного точки эллиптической кривой, рассматриваемой над конечным полем, имеет большую вычислительную сложность, чем проблема факторизации числа или проблема вычисления дискретного логарифма в полях той же размерности.

В начале 2000-х годов в криптографию вошло и получило большую популярность преобразование Вейля, которое позволило разработать и реализовать много новых алгоритмов типа *короткой* ЭЦП, ЭЦП, основанной на идентификационных данных пользователя, многосторонние протоколы Диффи-Хеллмана и другие.

В данном электронном учебнике мы постарались собрать учебный материал, относящийся к основам построения современных алгоритмов криптографии и соответствующий вспомогательный материал, связанный с этими алгоритмами. В частности, мы приведем сведения из теории групп, колец, конечных полей, теории чисел.

Дополнительные сведения можно получить из монографий А.В.Черемушкина «Лекции по арифметическим функциям в криптографии»,

МЦНМО, 2002 [85] и О.Н.Василенко «Теоретико-числовые алгоритмы в криптографии», МЦНМО, 2003 [51] и других источников, упомянутых в списке литературы в конце учебника.

Глава 1

Введение в информационную безопасность.

1.1 Основные понятия информационной безопасности

Информационная безопасность – это состояние информационной системы, в котором угрозы нарушения конфиденциальности, целостности и доступности информации сведены к минимуму.

Разберем основные составляющие, на которые раскладывается задача обеспечения и поддержки информационной безопасности.

Конфиденциальность – это свойство информации, обеспечивающее ее адресный доступ к пользователям информационных систем. Иначе говоря, информация должна быть доступна зарегистрированным легальным пользователям, имеющим разрешение владельца на доступ к информационному ресурсу, и недоступна всем остальным несанкционированным пользователям.

Целостность – это свойство информации, обеспечивающее ее неизменность или изменение только в соответствие с точными правилами, установленными владельцем информации.

Доступность – это это свойство информации, обеспечивающее право легальных пользователей на неограниченный доступ к всем ресурсам информационной системы или их части.

В зависимости от типа информационного ресурса или условий его функционирования на первый план может выступать та или иная составляющая информационной безопасности.

Например, если речь идет о коммерческих разработках, имеющих материальную ценность вследствие вложенных в них денег, времени и интеллектуальных усилий, то наиболее важным является сохранить их

в тайне от конкурентов, т.е. обеспечить конфиденциальность ресурсов. Также конфиденциальными являются данные банковских карт, сведения о личной жизни, состоянии здоровья людей и другие сведения, принадлежащие группам людей.

Другие сведения не представляют собой какой-либо тайны, или, наоборот, должны быть *доступны* всем и доставляться всевозможными способами, например, сведения об опасности, подстерегающих путешественников в той или иной местности, возможных болезнях, негативных последствиях применения наркотиков или медикаментов. Люди, скрывающие такие сведения, являются преступниками и должны быть подвергнуты наказанию.

В третьих случаях необходимо обеспечить защиту информации от незаконного изменения. Таковой является, например, информация о результатах сдачи абитуриентами ЕГЭ по математике или результаты любых других экзаменов. Правила выставления оценок по ЕГЭ строго регламентированы, и изменение допустимо только в ограниченных случаях в строго определенной последовательности. Это свойство информации мы называем целостностью.

Средства информационной безопасности должны обеспечивать защиту всех указанных свойств информации с использованием комплекса методов и средств защиты.

1.2 Методы информационной безопасности.

Защита информации обеспечивается различными методами и средствами. В целом, их можно разбить на три большие группы: физические, организационно-правовые и технические средства защиты.

Физические методы защиты образуют внешний уровень защиты. Сюда можно отнести, например, службу охраны учреждения, проверяющей документы на входе в помещение, где содержатся информационные ресурсы (первичная авторизация, разделяющая толпу на людей, имеющих право на доступ к ресурсам, и на граждан, не имеющих таких прав). В эту группу методов защиты также относятся меры защиты от пожаров, скачков напряжения электропитания, наводнений и других техногенных опасностей. Всевозможные датчики, видеокамеры, электронные замки, установленные на охраняемых помещениях, ограничивают перемещение персонала в защищаемых местах и уменьшают вероятность проникновения потенциального нарушителя.

Организационно- правовые методы предназначены для формирования общей политики безопасности учреждения, подбора кадрового состава и включают воспитательные меры защиты, специальные процедуры принятия и увольнения сотрудников (именно обиженные сотрудники представляют особую угрозу для безопасности предприятия в силу знания специфики работы и структуры защиты). Воспитательные меры призваны также воспитывать у сотрудников чувство ответственности за свою работу, работу коллектива и предприятия, в целом, а также информировать служащих о мерах наказания, предусмотренных за те или иные нарушения.

Большую подгруппу в группе организационно- правовых методов образуют законодательные меры и методы защиты, среди них – федеральные законы и указы, региональные, отраслевые законы и постановления, правила внутреннего распорядка и поведения людей в помещениях, содержащих информационные ресурсы и т.п. Кроме знания тех ограничений, которые возникают при работе с защищаемой информацией, необходимо обучать персонал правильной работе, т.к. неправильное обращение с компонентами информационной системы может нанести вред не меньший, чем умышленное повреждение информационного ресурса, которое встречается сравнительно реже.

К техническим методам защиты отнесем программно- аппаратные средства, осуществляющие процедуры аутентификации пользователей, защиту данных от несанкционированного доступа и чтения (криптографические средства), а также средства безопасной передачи данных по сетям, защиту от вирусов и т.п.

Отметим, что вышеупомянутые методы защиты принято делить также дополнительно на предупреждающие, выявляющие и восстанавливающие методы защиты.

Первые подобно законам и постановлениям предупреждают взломщика о потенциальных мерах наказания, предусмотренных за те или иные нарушения.

Вторые подобно датчикам и сигнализаторам выявляют наличие нарушений в сфере информационной безопасности и предупреждают владельца информации или службу безопасности о вторжении в защищаемую структуру.

Третьи служат для восстановления ущерба, нанесенного при атаке на информационные ресурсы или службы защиты. Эти три дополнительные три группы можно найти как среди физических, так и среди

организационных и технических средств и методов защиты.

Например, при работе с базами данных ограничение доступа персонала к помещению, где находится сервер баз данных, является предупреждающей мерой защиты, аутентификация и аудит пользователей БД – выявляющей мерой, а архивирование данных и восстановление после сбоя восстанавливающей мерой защиты.

Не умаляя значимости физических и организационно- правовых методов защиты, мы основное внимание уделим техническим методам защиты, в группу которых включается и криптография.

1.3 Сервисы информационной безопасности.

Сервисы информационной безопасности – это основные комплексы защитных средств, предназначенные для поддержки безопасности системы. К этим сервисам относятся *аутентификация, авторизация и аудит.*

Соответствующие английские написания этих терминов имеют вид: authentication, authorization, audit, и начинаются с букв Au, обозначающих химический элемент "золото" в периодической таблице Менделеева. Поэтому сервисы аутентификации, авторизации и аудита называют *золотыми правилами* информационной безопасности. Дадим краткое определение этих терминов.

Аутентификация

Аутентификация – это процедура проверки идентификатора пользователя. При входе в информационную систему пользователь должен идентифицировать себя, т.е. отождествить себя с одним из зарегистрированных в системе пользователей. Для этого в самой распространенной системе парольной аутентификации пользователь вводит свои логин и пароль.

Логин является идентификатором пользователя. Наличие пользователя с таким идентификатором проверяется по базе данных зарегистрированных пользователей. Однако, идентификаторы пользователей не являются секретными данными, поэтому кто-угодно может оказаться под предъявляемым логином. Значит, необходимо проверить тождественность идентификатора субъекту, пытающемуся войти в систему. Для этого и служит пароль, который и является тем секретом, который должен знать только сам пользователь и система аутентификации, проверяющая легальность пользователя.

Авторизация

Авторизация – это процедура разделения пользователей на группы с разными правами доступа.

Зарегистрированный пользователь, входящий в систему, не обязательно получит полные права на доступ к системе. Для того, чтобы точно определить кому какие права на и на какие объекты передать используется авторизация. В результате выполнения этой процедуры некоторые пользователи получат полный доступ к системе, некоторые - только право на чтение, а некоторые - ограниченные права по доступу к специализированному набору сервисов.

Например, представим себе работу деканата учебного заведения по регистрации текущих оценок студентов. Есть разные группы пользователей, получающие доступ к оценкам:

1. Сами студенты и их родители могут читать данные, относящиеся к группе, в которой студент занимается.

2. Преподаватель по предмету может читать и изменять оценки по своему предмету, но только в определенные временные интервалы.

3. Замдекана проверяет оценки, введенные преподавателем, и фиксирует их в окончательной таблице данных, после чего изменение становится невозможным или возможным с разрешения декана факультета.

Авторизация – непременный атрибут всех систем управления базами данных, в которых эта процедура может тщательно прописана для каждого объекта базы данных: схемы таблицы, данных в таблице, отдельных элементов, запросов (статистики), форм, отчетов, макросов и программных процедур.

Аудит

Аудит – это процедура записи действий всех пользователей по доступу к защищаемым данным.

Если какой-то пользователь неправомерно использует доступ к системе для расширения своих полномочий или пытается нарушить нормальное функционирование системы путем, например, подбора паролей, выполнения SQL-инъекций или других незаконных действий, то аудит позволить определить виновного и передать данные администратору системы. В большинстве случаев правильно настроенная ИС сама автоматически определит попытки взлома и блокирует нападавшего.

1.4 Угрозы информационной безопасности

Организация надежной защиты не может быть выполнена без анализа *угроз* ИБ и степени их опасности. Под угрозой понимается потенциально возможное событие, которое может привести к нанесению вреда информационной системе и хранящейся в ней информации.

Существует различные классификации угроз. Приведем здесь некоторые из таких классификаций (см. Шаньгин Ф.Ф. Защита компьютерной информации: эффективные методы и средства [86]):

1. По природе возникновения угрозы делятся на
   - естественные угрозы, вызванные естественными физическими процессами или природными явлениями (наводнениями, землетрясениями, сбоями электропитания и т.п);
   - искусственными угрозами, вызванными деятельность человека (некомпетентные действия персонала, действия хакеров, злоумышленные действия конкурентов и т.п.);
2. По степени преднамеренности угрозы различаются как
   - угрозы, вызванные ошибками или халатностью персонала, например, неправильное использование информационной системой;
   - угрозы преднамеренного действия, например, действия злоумышленников.
3. По местоположению угрозы могут быть:
   - вне зоны расположения ИС;
   - в пределах контролируемой зоны ИС;
   - непосредственно в ИС, например, зараженные вирусами программные средства.
4. По степени вреда, наносимого информационной системе:
   - несущественное, легко восстановимое нарушение работы ИС;
   - существенное нарушение работы ИС, требующие серьезных мер по восстановлению нормальной работы;
   - критическое воздействие на работу ИС, вызвавшее полное крушение системы, или важных компонент ее работы, потеря жизненно важной информации, хранящейся в системе, блокирование

каналов связи, требующее длительного времени восстановления и другие угрозы, наносящие существенный вред информационной системе.

1.5 Классификация критографических методов защиты информации

Криптографические методы защиты информации входят в систему технических методов защиты. Эти методы делятся на следующие группы:

1. Методы шифрования информации, предназначенные для защиты информации от чтения, с помощью криптографического преобразования, превращающего исходный текст в нечитаемую последовательность символов;
2. Методы построения электронной цифровой подписи (ЭЦП), предназначенные для защиты информации от изменения, связывания источника информации (автора) с самой информацией;
3. Методы хеширования данных, предназначенные для проверки целостности электронных документов без чтения самой информации в этих документах;

Классические методы шифрования и построения ЭЦП включают в себя сдвиги и перестановки исходного текста, гаммирование (наложение одного текста на другой), подстановки (отображения одних алфавитов на другие) или комбинации этих методов. Например, такой классический шифр как DES (Data Encryption Standard), разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт, имеет блоки по 64 бита и использует ключ длины 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований.

В нашем учебнике мы рассмотрим неклассические системы, которые появились примерно в середине 70-х годов XX столетия.

1.6 Особенности систем с открытым ключом

Шифрование – это процедура преобразования произвольного электронного файла (текстовой строки) в нечитаемую последовательность символов для защиты файла от чтения людьми, не имеющими соответствующих прав доступа.

Ключом шифрования (построения электронной цифровой подписи ЭЦП) называется произвольный конечный набор символов некоторого алфавита, который служит входным параметром для алгоритма шифрования (построения ЭЦП).

Классические системы шифрования такие, как шифры перестановки, подстановки (например, DES и RC4) используют один и тот же ключ как для шифрования, так и для расшифрования текстов.

Система RSA относится к *системам шифрования с открытым ключом*. Это означает, что для шифрования и расшифрования используются совершенно разные ключи. Ключ шифрования, используемый для преобразования исходного текста в шифротекст, разрешается передавать любому пользователю системы, не задумываясь о безопасности, поэтому этот ключ называется *открытым – public* в отличии от второго *секретного– private* ключа, который используется для восстановления исходного текста. Это позволяет свободно передавать и обменивать ключи по различным сетям с открытым доступом.

Глава 2

Вспомогательные сведения из теории алгоритмов

Теория сложности вычислений является разделом теории вычислений, изучающим сложность(стоимость) работы, требуемой для решения вычислительной проблемы.

Сложность обычно измеряется абстрактными понятиями времени и пространства, называемыми вычислительными ресурсами.

Время определяется количеством тривиальных шагов, необходимых для решения проблемы, тогда как пространство определяется объёмом памяти или места на носителе данных или количеством процессоров для обработки данных.

2.1 Основные понятия теории сложности

Одним из наиболее важных понятий математики является понятие алгоритма.

Определение. Алгоритм – это конечный набор инструкций для решения класса однотипных задач.

Отметим наиболее важные характеристики алгоритма:

- *Конечность.* Алгоритм состоит из конечного числа инструкций.

- *Наглядность.* Каждая инструкция представляет собой простое действие.

- *Массовость.* Алгоритм применим не к одной задаче, а к классу задач, давая для каждой задачи из области определения свое решение.

- *Детерминированность.* Действие алгоритма однозначно по отношению к одним и тем же входным данным независимо от условий выполнения алгоритма, текущего времени или других условий.

Впрочем, есть класс недетерминированных алгоритмов, в которых последнее условие не выполняется. В таких алгоритмах выполнение вычисления можно выполняться одновременно по нескольким альтернативным путям, приводящим к разным ответам.

Задачи теории сложности

Основными задачами теории сложности алгоритмов являются следующие:

- Обеспечить методику количественной оценки сложности проблемы в абсолютных терминах.
- Дать метод сравнения сложности двух различных проблем.
- Дать строгое определение эффективного алгоритма.

Размер проблемы

Под размером проблемы понимается длина входных данных в том или ином представлении. Обычно длина входных данных измеряется в битах. Например, длина ключа RSA равна 1024 или 2048 битам. Представление данных в разных системах может иметь разную длину, однако разные представления отличаются обычно в константное число раз. Например, представление ключей RSA в десятичном представлении имеет длину, меньшую битового, в $log_2 10$ раз.

Замечание. Некоторые представления задачи могут иметь существенно меньшую длину. В алгоритмической теории информации определяется колмогоровская сложность объекта как мера вычислительных ресурсов, необходимых для точного определения этого объекта. Некоторые ресурсы по этой мере могут иметь существенно меньшую длину представления. Следует отметить, однако, представляя данные в сокращенной форме, можно получить дополнительные издержки при выполнении алгоритма.

Оценка скорости работы алгоритма

Существуют *временная* и *пространственная* сложность алгоритма.

Временная сложность оценивает время или количество тактов работы алгоритма относительно длины входных данных. Например, время работы алгоритма сортировки n-мерного числового массива методом пузырька оценивается сверху величиной $C \cdot n^2$ для некоторой константы $C > 0$. Длина входных данных здесь равна $q\,n$, где q – длина представления элемента массива.

Поскольку заданную длину могут иметь много входных задач, то временная оценка строится для всех задач из класса задач с входными данными одинаковой длины. Поэтому могут существовать наилучшая, средняя и наихудшая оценка времени работы алгоритма.

- *Наилучший случай.* Оценивает время работы самой простой задачи из класса задач заданной длины. Дает мало информации о параметрах сложности.

- *Средний случай.* Оценивает вероятностное распределение параметров сложности.

- *Наихудший случай.* Облегчает анализ проблемы.

Пространственная сложность оценивает количество памяти, требуемой для выполнения вычисления. Эти две меры тесно связаны. Обычно, алгоритмы, требующие много времени вычисления, потребляют много памяти, но бывают и исключения. Некоторые алгоритмы - требовательны к памяти, и при нехватке памяти работают значительно медленнее.

Временная сложность (асимптотика)

Для сравнения алгоритмов используются специальные обозначения.

Обозначение	Интуитивное объяснение	Определение
$f(n) \in O(g(n))$	f ограничена сверху функцией g с точностью до постоянного множителя асимптотически	$(\exists C)(\exists n_0)(\forall n \geq n_0)$ $f(n) \leq C \cdot g(n)$
$f(n) \in \Omega(g(n))$	f ограничена снизу функцией g с точностью до постоянного множителя асимптотически	$(\exists C)(\exists n_0)(\forall n \geq n_0)$ $f(n) \geq C \cdot g(n)$
$f(n) \in \Theta(g(n))$	f ограничена сверху и снизу функцией g с точностью до постоянного множителя асимптотически	$(\exists C_1)(\exists C_2)(\exists n_0)(\forall n \geq n_0)$ $C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$
$f(n) \in o(g(n))$	g доминирует над f асимптотически	$\lim_{n\to\infty} f(n)/g(n) = 0$
$f(n) \in \omega(g(n))$	f доминирует над g асимптотически	$\lim_{n\to\infty} f(n)/g(n) = \infty$
$f(n) \sim (g(n))$	f эквивалентна g асимптотически	$\lim_{n\to\infty} f(n)/g(n) = 1$

Критерии оценки сложности (1)

- Если создаваемая программа будет использована только несколько раз, тогда стоимость написания и отладки программы будет доминировать в общей стоимости программы. В этом случае следует предпочесть алгоритм, являющийся наиболее простым для реализации.
- Если программа будет работать только с "малыми" входными данными, то степень роста времени выполнения будет иметь меньшее значение, чем константа, присутствующая в формуле времени выполнения. Вместе с тем и понятие «малости» входных данных зависит от точного времени выполнения конкурирующих алгоритмов. Существуют алгоритмы, такие как алгоритм целочисленного умножения, асимптотически самые эффективные, но которые никогда не используют на практике даже для больших задач, так как их константы пропорциональности значительно превосходят подобные константы других, более простых и менее "эффективных" алгоритмов.
- Эффективные, но сложные алгоритмы могут быть нежелательными, если готовые программы будут поддерживать лица, не участвующие в написании этих программ.
- Известно несколько примеров, когда эффективные алгоритмы требуют таких больших объемов машинной памяти (без возможности использования более медленных внешних средств хранения), что этот фактор сводит на нет преимущество «эффективности» алгоритма.
- В численных алгоритмах точность и устойчивость алгоритмов не менее важны, чем их временная эффективность.

2.2 Некоторые важные классы алгоритмов

Определение. Функция $f(n) = n^k$, где $k > 0$, называется полиномиальной. Функция $f(n) = a^n$, где $a > 1$, называется экспоненциальной функцией или простой экспонентой.

Полиномиальные алгоритмы

Определение. Алгоритм, временная сложность которого ограничена полиномиальной функцией, называется *полиномиальным алгоритмом.*

Пример: Нахождение кратчайшего пути в графе с неотрицательным весом. *Сложность:* $O(n^2)$.

Другие примеры: сортировка, поиск во множестве, выяснение связности графов.

Экспоненциальные алгоритмы

Определение. Алгоритм, временная сложность которого ограничена экспоненциальной функцией, называется *экспоненциальным алгоритмом.*

Пример: Выбор n-разрядного числа перебором. Сложность: $O(10^n)$.

Другие примеры: задача коммивояжера, задача выполнимости булевых формул.

Субэкспоненциальные алгоритмы

Определение. Алгоритм, временная сложность которого меньше экспоненциальной, но не является полиномиальной.

Пример: Разложение n-разрядного числа на простые множители (факторизация n) методом решета числового поля

$$\text{Сложность: } O(e^{C(\ln\ln n)^{1/3}(\ln n)^{2/3}})$$

Другие примеры: Вычисление дискретного логарифма в конечном поле.

2.3 Модели вычислений

Для формального описания алгоритма в 30-е годы XX столетия было предложено несколько моделей вычислений. Самая известная из них – машина Тьюринга.

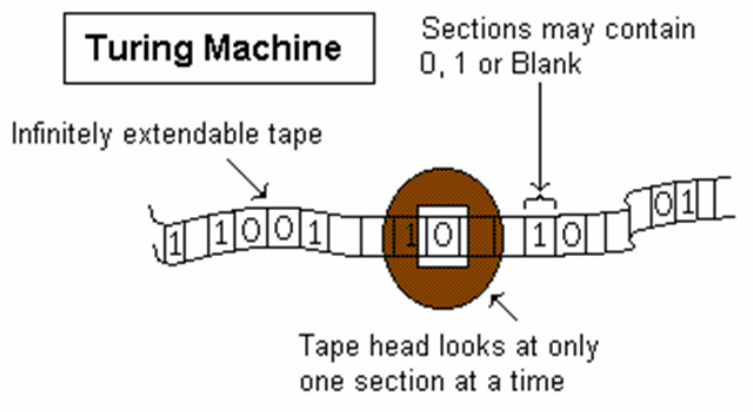


Рис. 2.1: Машина Тьюринга

Машина Тьюринга

Машина Тьюринга – абстрактная вычислительная машина, предложенная английским математиком Аланом Тьюрингом в 1936 году для формализации понятия алгоритма.

В состав машины Тьюринга (МТ) входит бесконечная в одну сторону лента (в некоторых описаниях - бесконечная в обе стороны лента), разделённая на ячейки, и читающее - записывающее устройство (ЧЗУ). Каждая ячейка на ленте может содержать либо пустой символ λ, либо один из символов конечного алфавита $A = \{a_0, a_1, \ \ldots, a_n\}$ (обычно алфавит A состоит из 0 и 1).

МТ работает по тактам, начиная с такта 0. Каждый такт МТ находится в одном из внутренних состояний множества $Q = \{q_0, q_1, \ \ldots, q_k\}$, причем состояние q_1 является начальным, а состояние q_0 - конечным (финальным).

ЧЗУ каждый момент времени обозревает одну ячейку ленты, может перемещаться на одну клетку влево и вправо или остаться на месте, читать и записывать в ячейки ленты символы алфавита A.

В начальный момент работы МТ на ленте записано входное слово символами алфавита A, начиная с клетки ленты с номером 0, ЧЗУ читает содержимое клетки 0, и МТ находится в состоянии q_1.

Пустой символ заполняет все клетки ленты, кроме конечного числа, на которых записаны входные данные. Управляющее устройство работает согласно правилам перехода, которые представляют алгоритм, реализуемый данной машиной Тьюринга. Каждое правило перехода предписывает машине, в зависимости от текущего состояния и наблюдаемого

в текущей клетке символа, записать в эту клетку новый символ, перейти в новое состояние и переместиться на одну клетку влево, вправо или остаться на месте.

Вид команды машины Тьюринга:

$$< \mathbf{q_i}, \mathbf{a_i} \ \rightarrow \mathbf{q_j}, \ \mathbf{a_j}, \ \mathbf{s_j}, >, \quad s_j \in \{L, R, S\}.$$

Эта команда выполнится в момент времени t, если в этот момент машина находится во внутреннем состоянии q_i, и читающее устройство обозревает ячейку, в которой записан символ a_i.

При выполнении команды, машина перейдет в новое состояние q_j, заменит символ a_i на a_j и передвинет читающее устройство на одну клетку влево, вправо или оставит его в той же позиции в зависимости от значения s_j.

Машина Тьюринга работает до тех пор, пока не перейдет в финальное состояние q_0. Выходным значением (результатом работы) машины станет число y, равное общему количеству единиц на ленте в момент остановки машины T.

Машина Тьюринга называется детерминированной, если каждой комбинации состояния и ленточного символа в таблице соответствует не более одного правила. Если существует несколько команд с одинаковой левой частью – входной парой «состояние – ленточный символ», то такая машина Тьюринга называется недетерминированной. Если в какой-то момент времени появляется две или более команды с одинаковой левой частью, соответствующей текущей конфигурации машины Тьюринга, вычисление разделяется на несколько ветвей, соответствующих разным командам с одинаковой левой частью. Каждая ветвь продолжает свое выполнение в параллельном режиме. Результатом вычисления недетерминированной машины Тьюринга полагается результат, полученный на той ветви, которая закончит свое вычисление раньше остальных.

Поскольку, число параллельный ветвей никак не ограничено, то, очевидно, что никакой реальный компьютер не может служить реализацией недетерминированной МТ. Однако, можно моделировать вычисление, выделяя каждому процессу некоторые ресурсы последовательно. В этом случае, работы машины Тьюринга будет выполнена, хотя вычисление продлится несколько дольше.

Отметим, что на некоторых входных аргументах машины Тьюринга могут выдавать бесконечное вычисление, не переходя никогда в финальное состояние. В таких случаях говорят, что машина Тьюринга *за-*

цикливается на аргументе x. Функция, вычислимая на таких машинах, является частично определенной. Возможность реализации частичных функций является важной характеристикой машин Тьюринга.

Класс функций, вычислимых на м.Тьюринга. Рекурсивные функции

Хотя машины Тьюринга представляют собой весьма примитивные вычислительные модели, в ходе обширных исследований было выяснено, что *любой* известный человечеству алгоритм можно реализовать на какой-нибудь машине Тьюринга, т.е. машина Тьюринга является универсальным вычислительным устройством, позволяющим реализовывать сколь угодно сложные алгоритмы.

Немецкий математик Курт Гедель в 1931 году определил и исследовал другую модель алгоритмов – алгоритмов, реализующих *рекурсивные функции*. Эти функции строятся как комбинации трех базовых операторов суперпозиции, минимизации и примитивной рекурсии, примененных к простых базовым функциям (функции, тождественно равной 0, функции прибавления 1 к аргументу, и набора функций выборки, выбирающих из n-и входных аргументов какой-нибудь один). Алгоритм вычисления каждой такой функции заложен в самом описании этой функции, поэтому каждая рекурсивная функция является *алгоритмически вычислимой*.

Дадим точное определение рекурсивной функции:

Базовыми рекурсивными функциями являются:
1. 0-местная функция (константа) O – ноль,
2.Функция следования $s(n) = n + 1$,
3.Функции выборки I_n^k : $I_n^k(x_1, x_2, \ldots, x_n) = x_k$, $0 < k \leq n$.

Оператором суперпозиции называется оператор подстановки одних функции в качестве аргументов других: пусть даны функции $f(x_1, x_2, \ldots, x_n)$, $g_1(y_1, \ldots, y_k)$, $\ldots$, $g_n(y_1, \ldots, y_k)$. Результатом суперпозиции будет функция $z = f(g_1, g_2, \ldots, g_n)[y_1, \ldots, y_k]$

Оператором примитивной рекурсии называется оператор, позволяющий из n-местной функции g и $(n+2)$-местной функции h строить $n+1$-местную функцию f по следующим правилам:

$$f(x_1, \ldots x_n, 0) = g(x_1, \ldots x_n)$$
$$f(x_1, \ldots x_n, k+1) = h(x_1, \ldots x_n, k, f(x_1, \ldots x_n, k))$$

(для вычисления $f(x_1, \ldots x_n, k+1)$ в h было подставлено значение f в предыдущей точке k).

Определение. Любая функция, которую можно получить из базовых функций применением конечного числа операторов суперпозиции и примитивной рекурсии, называется *примитивно-рекурсивной.*

Оператором минимизации называется оператор, строящий по $(n+1)$-местной функции g n-местную функцию f по следующему правилу:

$$(\forall x_1, \ldots, x_n)\ f(x_1, \ldots, x_n) = \min t\ [g(x_1, \ldots, x_n, t) = 0].$$

Примеры примитивно-рекурсивных функций:

1. $f(x, y) = x + y$
2. $f(x, y) =$ Н.О.Д. (x, y),
3. $f(x, y, z) = x^{y^z}$.

Определение. Любая функция, которую можно получить из базовых функций применением конечного числа операторов суперпозиции примитивной рекурсии и минимизации, называется *рекурсивной.*

Таким образом, все примитивно-рекурсивные функции являются рекурсивными. Обратное не верно. Важное отличие рекурсивных функций от примитивно-рекурсивных состоит в том, что первые являются всюду определенными в то время, как вторые могут быть и не всюду определенными.

2.4 Тезис Черча

Дальнейшие исследования показали, что класс функций, вычислимых на машинах Тьюринга, и класс рекурсивных функций совпадают.Более того, и другие подобные модели (нормальные алгорифмы Маркова, машины Поста, RAM-машины и др.) давали тот же класс функции, который покрывал все известные арифметические функции. Таким образом, можно было предположить, что все известные человечеству алгоритмы можно реализовать в рамках класса рекурсивных функций. Данное предположение (гипотеза, т.к. точно доказать ее невозможно) получила название *тезиса Черча* по имени английского математика Аллоиза Черча.

Тезис Черча: Любая *алгоритмически вычислимая* функция вычислима на какой-нибудь машине Тьюринга.

Формулировка тезиса Черча явилось одним из величайших достижений теории алгоритма, поскольку позволило дать формализацию (точное математическое описание) понятию *алгоритмически разрешимой* функции, как рекурсивной функции. Иначе говоря, если математическая задача не могла быть разрешена с помощью машины Тьюринга, то она не имела никакого решения вообще, т.е. являлась *алгоритмически неразрешимой.*

Конечные автоматы

Полезным инструментом для реализации математических алгоритмов явились *конечные автоматы (КА)* Также как и машина Тьюринга КА имеет конечное множество внутренних состояния и бесконечную ленту.

Главное отличие конечного автомата от машины Тьюринга – это линейность работы. Автомат работает по тактам, считывая конечное слово на входной ленте по одному символу за такт работы. Читающее устройство движется только в одном направлении, и работа закончится, когда будет пройдено входное слово. Поэтому время работы автомата всегда равно длине входного слова.

Каждый момент времени автомат находится в одном из конечного множества внутренних состояний. Множество внутренних состояний автомата Q делится на два непересекающихся подмножества Q_1 и Q_2. Если при завершении работы состояние автомата принадлежит Q_1, то говорят, что входное слово принимается данным автоматом. Иначе, входное слово *отвергается* автоматом.

Таким образом, конечный автомат выполняется разбиение множества входных слов на два непересекающихся подмножества: подмножество слов, принимаемых автоматом, и подмножество слов, отвергаемых автоматом.

2.5 Решающие проблемы и классы сложности

Решением алгоритмической проблемы (задачи) является, как правило, числовой ответ. Однако, есть класс проблем, решением которых является лишь константное значение **ДА** или **НЕТ**.

Пример 1. *Совместность системы уравнений*: для заданной системы уравнений относительно n неизвестных, определить, существует ли решение?

$$\begin{cases} x_1 + x_2^2 = 0 \\ x_1^3 - x_2 = 0 \\ x_1 + x_2 = 0 \end{cases}$$

Пример 2. *Проверка простоты натурального числа.* Для заданного натурального числа определить, является ли оно простым или составным.

Сводимости

Сводимость – это процедура, сводящая решение задачи из одного класса к решению из другого класса. Сводимость ранжирует классы алгоритмов по их сложности: тот класс, к которому сводятся другие классы, является более трудновычислимым. Дадим формальное определение сводимости.

Определение. Проблема P_1 сводится к проблеме P_2 если существует эффективная функция (алгоритм), позволяющая для каждого экземпляра Z_1 проблемы P строить экземпляр Z_2 проблемы P_2, так что задача Z_1 имеет решение "Да" тогда и только тогда, когда такое же решение имеет задача Z_2.

Обозначение: $P_1 \leq P_2$.

Полиномиальная сводимость

Проблема P_1 полиномиально сводится к проблеме P_2 если проблема P_1 сводится к проблеме P_2, и сводящая функция (алгоритм) полиномиально вычислима.

Обозначение: $P_1 \leq_p P_2$.

Пример. "Задача коммивояжера" полиномиально сводится к задаче "Выполнимость булевой функции".

Отметим, что если выполнено $P_1 \leq_p P_2$, то из существования полиномиального по времени алгоритма для P_2 следует существование полиномиального временного алгоритма для P_1.

Классы сложности

Определение. Класс P содержит все решающие проблемы, для которых существует алгоритм для машины Тьюринга, который приводит к правильному «да/нет» ответу за число шагов, ограниченному полиномом от размера задачи.

Определение. Класс NP содержит все решающие проблемы, для которых, имея некоторый ответ, существует полиномиальная проверка (сертификат) С того, что данный ответ дает правильное «да/нет» решение проблемы.

Определение. Класс $co-P$ содержит все решающие проблемы, для которых существует полиномиальный алгоритм, который проверяет, какие из «да/нет» ответов неверны.

Определение. Класс $co-NP$ содержит все решающие проблемы, так что существует полиномиальная проверка (сертификат) C , которая определяет, верно ли, что проблема не имеет правильных «да/нет» ответов.

Важнейшие гипотезы

Хотя определения классов алгоритмов P, $co-P$, NP и $co-NP$ интуитивно понятны, важнейшие утверждения относительно непустоты классов разностей $co-NP\ -\ P$, $NP\ -\ P$ и $NP\ -co-P$ не доказаны.

Впервые вопрос о равенстве классов P и NP был поставлен Стивеном Куком (S.A.Cook) в 1971 году. Этот вопрос является одной из центральных открытых проблем уже более 40 лет. В теории алгоритмов вопрос о равенстве классов сложности P и NP является одной из центральных открытых проблем. Положительный ответ будет означать, что существует (по-крайней мере, теоретическая возможность) решать многие сложные задачи существенно быстрее, чем сейчас. Однако, вероятность такого исхода практически равна нулю. Для отрицательного решения необходимо построить пример алгоритмической задачи, решаемой на какой-то недерминированной м.Тьюринга, которую невозможно решить ни на какой детерминированной м.Тьюринга.

Проблема $\mathbf{P = NP}$ является одной из семи задач тысячелетия, за решение которой Математический институт Клэя назначил премию в миллион долларов США. Основными гипотезами, которых придерживается большинство математиков, являются следующие:

- $P = co-P$

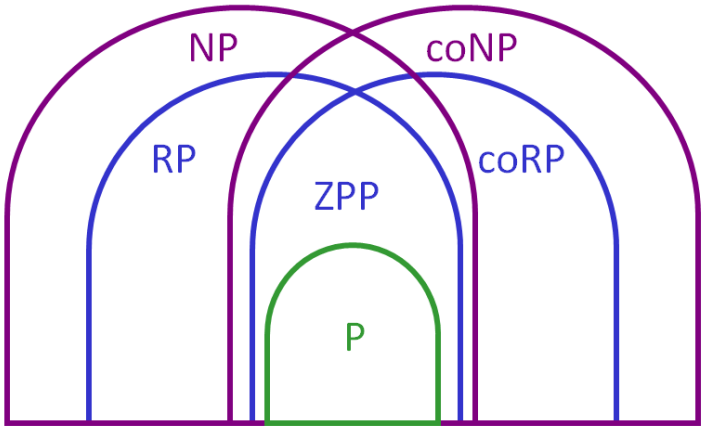


Рис. 2.2: Соотношения классов сложности

- $NP \neq co - NP$
- $P \neq NP$

NP-полные проблемы

Определение. (NP-полнота) *NP-полная решающая проблема* Z – это такая задача из класса NP, к которой можно свести любую другую решающую проблему Z' из класса NP, то есть, $Z' \leq_p Z$.

Таким образом, NP-полные задачи образуют в некотором смысле подмножество «самых сложных» задач в классе NP; и если для какой-то из них будет найден «быстрый» алгоритм решения, то и любая другая задача из класса NP может быть решена так же «быстро». В настоящее время доказано, что NP-полными являются многими известные математические задачи:

1. Задача SAT проверки выполнимости произвольно булевой формулы (достаточно ограничиться формулами от трех переменных).
2. Задача коммивояжера.
3. Проблема раскраски графа.
4. Задача о вершинном покрытии.

Как избежать NP-полноты?

- Попытаться доказать $P = NP$ (приз - 1 млн.долларов).
- Использовать рандомизированные алгоритмы.
- Использовать приближенные алгоритмы.

- Использовать эвристические алгоритмы.
- Использовать квантовые алгоритмы.

Глава 3

Основы теории кодирования

КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ - процессы представления информации в определенной стандартной форме и обратный процесс восстановления информации по ее такому представлению. В математической литературе кодированием называется отображение произвольного множества в множество конечных последовательностей (слов) в некотором алфавите , а декодированием - обратное отображение.

Примерами кодирования являются: представление натуральных чисел в r-й системе счисления, при котором каждому числу $N = 1, 2, ...l$ ставится в соответствие слово $b_1,\ b_2, ...b_l$ в алфавите , $0 \leq r \leq r-1$, такое, что $b_1 \neq 0$ и $b1 \cdot r^{l-1} + ... + b_{l-1} \cdot r + b_l = N$.

Примерами такого кодирования могут служить преобразования текстов на русском языке с помощью телеграфного кода в последовательности, составленные из посылок тока и пауз различной длительности; отображение, применяемое при написании цифр почтового индекса. В последнем случае каждой десятичной цифре соответствует слово в алфавите В $B_2 = \{0, 1\}$ длины 9, в котором символами 1 отмечены номера использованных линий (например, цифре 5 соответствует слово 110010011). Исследование различных свойств кодирований/декодирований и построение эффективных в определенном смысле кодирований, обладающих требуемыми свойствами, составляет проблематику теории кодирования. Обычно критерий эффективности кодирования так или иначе связан с минимизацией длин кодовых слов (образов элементов множества А), а требуемые свойства кодирования связаны с обеспечением заданного уровня помехоустойчивости, понимаемой в том или ином смысле. В частности, под помехоустойчивостью понимается возможность однозначного декодирования при отсутствии или допустимом уровне искажений в кодовых словах. Помимо помехоустойчивости, к кодированию может предъявляться ряд дополнительных требований. Например, при выборе кодирования

для цифр почтового индекса необходимо согласование с обычным способом написания цифр. В качестве дополнительных требований часто используются ограничения, связанные с допустимой сложностью схем, осуществляющих К. и д. Проблематика теории кодирования в основном создавалась под влиянием разработанной К. Шенноном (C. Shannon, [87]) теории передачи информации.

Основной литературой по данному разделу являются книги [45], [65], [69], [73] и [87].

3.1 Кодирование информации и шумы. Глоссарий.

Канал - среда передачи информации, например, телефонная линия или атмосфера

Кодирование источника – устранение «лишней», сжатие информации.

Кодирование канала – добавление избыточности для обнаружения и/или исправления ошибок (в результате шума) – защита от случайных воздействий.

Шум – фактор, влияющий на передачу информации. Шум может возникнуть из-за магнитной бури, молнии, метеоритного дождя, случайного искажения звука в радиопередаче, плохой печати изображения или текста, плохой слышимости.

В результате шума сообщение может исказиться.

3.2 Структура кодера и декодера

Преобразование дискретного сообщения в сигнал обычно осуществляется в виде двух операций - кодирования и модуляции. Кодирование представляет собой преобразование сообщения в последовательность кодовых символов.

Простейшим примером дискретного сообщения является текст. Любой текст состоит из конечного числа элементов: букв, цифр, знаков препинания. Их совокупность называется алфавитом источника сообщения. Так как число элементов в алфавите конечно, то их можно пронумеровать и тем самым свести передачу сообщения к передаче последовательности чисел.

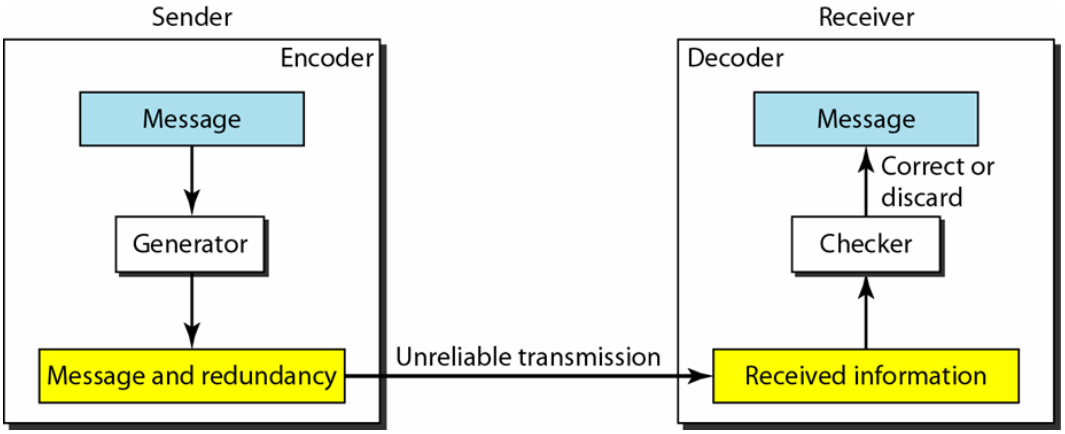


Рис. 3.1: Структура кодера и декодера

Так, для передачи букв русского алфавита (их 32) необходимо передать числа от 1 до 32. Для передачи любого числа, записанного в десятичной форме, требуется передача одной из десяти цифр от 0 до 9 для каждого десятичного разряда. На практике при кодировании дискретных сообщений широко применяется двоичная система счисления.

При кодировании происходит процесс преобразования элементов сообщения в соответствующие им числа (кодовые символы). Каждому элементу сообщения присваивается определенная совокупность кодовых символов, которая называется кодовой комбинацией. Совокупность кодовых комбинаций, обозначающих дискретные сообщения, образует код.

Правило кодирования может быть выражено кодовой таблицей, в которой приводятся алфавит кодируемых сообщений и соответствующие им кодовые комбинации. Множество возможных кодовых символов называется кодовым алфавитом, а их количество m - основанием кода.

В общем случае при основании кода m правила кодирования N элементов сообщения сводятся к правилам записи N различных чисел в m-й системе счисления. Число разрядов n, образующих кодовую комбинацию, называется значностью кода, или длиной кодовой комбинации. В зависимости от системы счисления, используемой при кодировании, различают двоичные и m-е (недвоичные) коды.

Коды, у которых все комбинации имеют одинаковую длину, называют равномерными. Для равномерного кода число возможных комбинаций равно $m \cdot n$. Примером такого кода является пятизначный код Бодо, содержащий пять двоичных элементов ($m = 2,\ n = 5$). Число возможных кодовых комбинаций равно $2^5 = 32$, что достаточно для кодирования

всех букв алфавита. Применение равномерных кодов не требует передачи разделительных символов между кодовыми комбинациями.

Неравномерные коды характерны тем, что у них кодовые комбинации отличаются друг от друга не только взаимным расположением символов, но и их количеством. Это приводит к тому, что различные комбинации имеют различную длительность. Типичным примером неравномерных кодов является код Морзе, в котором символы 0 и 1 используются только в двух сочетаниях - как одиночные (1 и 0) или как тройные (111 и 000). Сигнал, соответствующий одной единице, называется точкой, трем единицам - тире. Символ 0 используется как знак, отделяющий точку от тире, точку от точки и тире от тире. Совокупность 000 используется как разделительный знак между кодовыми комбинациями.

По помехоустойчивости коды делят на простые (примитивные) и корректирующие. Коды, у которых все возможные кодовые комбинации используются для передачи информации, называются простыми, или кодами без избыточности. В простых равномерных кодах превращение одного символа комбинации в другой, например 1 в 0 или 0 в 1, приводит к появлению новой комбинации, т. е. к ошибке.

3.3 Корректирующие коды

Одним из наиболее значимых методов борьбы с ошибками является *избыточное кодирование*. Целью избыточного кодирования является обнаружение и (или) исправление ошибок путем добавления дополнительной (избыточной) информации в код. С этой целью корректирующие коды строятся так, что для передачи сообщения используются не все кодовые комбинации m^n, а лишь некоторая часть их (так называемые разрешенные кодовые комбинации).

Декодирование состоит в восстановлении сообщения по принимаемым кодовым символам. Устройства, осуществляющие кодирование и декодирование, называют соответственно кодером и декодером. Как правило, кодер и декодер выполняются физически в одном устройстве, называемым кодеком.

Методы борьбы со случайными ошибками

При определении помехоустойчивости кодирования формализуется понятие ошибки и вводится в рассмотрение некоторая модель образования ошибок.

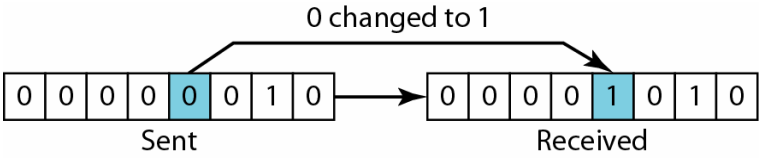


Рис. 3.2: Ошибка в одном разряде

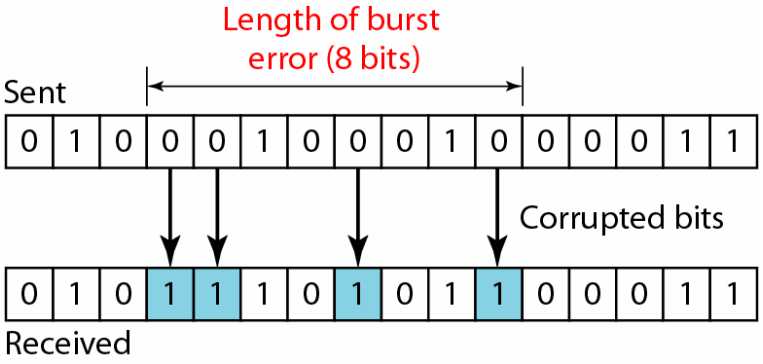


Рис. 3.3: Пакет ошибок длины 8

Ошибкой типа замещения (или просто ошибкой) называется преобразование слова, состоящее в замещении одного из его символов другим символом алфавита В рис.2 произошло замещение символа 0 символом 1. Возможность обнаружения и исправления ошибок основана на том, что для кодирования f, обладающего ненулевой избыточностью, декодирование f^{-1} может быть произвольным образом доопределено на r подсловах, не являющихся кодовыми. В частности, если множество разбито на m непересекающихся подмножеств $D_0, \ldots, D_{m-1}$ таких, что декодирование f^{-1} доопределено так, что $f^{-1}(D_i) = i$, то при декодировании будут исправлены все ошибки, преобразующие кодовое слово $f(i)$ в D_i, $i = 0, ..., m-1$. Аналогичная возможность имеется и в случае ошибок других типов таких, как стирание символа (замещение символом другого алфавита), изменение числового значения кодового слова на $b \in \{1, \ldots, r-1\}$ (арифметическая ошибка), выпадение или вставка символа и т. п.

Другие модели ошибок

Стирающий канал. Стирающие коды являются самыми «молодыми» среди всех известных ныне помехоустойчивых кодов, и их изученность на сегодня не является полной.

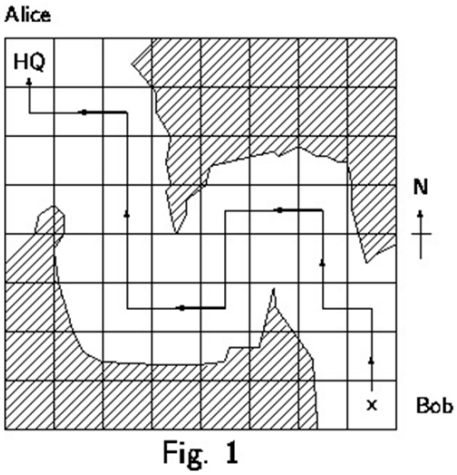


Рис. 3.4: Путь от Боба до Алисы

Согласно определению, *стирание* – это ошибка, позиция которой известна, а значение не определено.

Примером таких ошибок могут служить потери пакетов, имеющие место в сетях IP. Поскольку в сетях IP используется сквозная нумерация пакетов, то существует возможность определить позицию потерянного пакета. Значение стертого пакета неизвестно получателю (декодеру).

Канал со вставками. Метод использует дополнительные биты для вставки контрольной информации.

3.4 Пример избыточного кодирования.

Предположим, что требуется закодировать путь от Боб до Алисы (рис.4).

Обозначим один шаг в направлении "Север", "Запад", "Юг", "Восток" кодами "N", "W", "S", "E" соответственно. Тогда весь путь изобразится последовательностью

$$S_{A,\,B} = \{N,\ N,\ W,\ N,\ N,\ W,\ W,\ S,\ S,\ W,\ W,\ N,\ N,\ N\ \}$$

1. **Неизбыточное кодирование.** Будем кодировать направления движения двухбитовыми последовательностями:

$$\{00, 01, 10, 11\}.$$

Очевидно, что любая ошибка в одном бите приводит к потере информации.

2. **Избыточное кодирование.** Будем кодировать направления движения трехбитовыми последовательностями:

$$\{000, 011, 101, 110\}.$$

Теперь изменение одного бита позволяет определить *наличие ошибки.*

3. **Избыточное кодирование 2.** Будем кодировать направления движения пятибитовыми последовательностями:

$$\{00000, 01101, 10110, 11011\}.$$

Теперь ошибка в одном бите не только будет обнаружена, но и ее можно будет исправить.

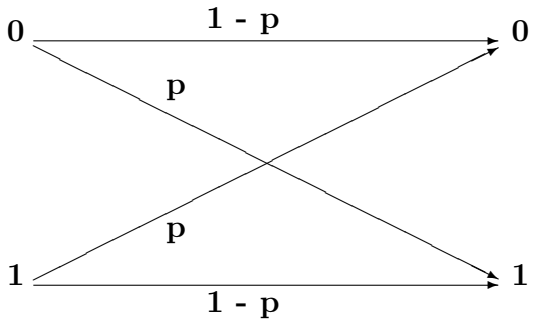
Цели передачи по каналу с шумом

1. Быстрое кодирование информации.
2. Простой способ передачи закодированного сообщения.
3. Быстрое декодирование полученной информации.
4. Надежная очистка от шума.
5. Передача максимального объема информации в единицу времени.

ДСК – двоичный симметричный канал

Двоичность: Алфавит ={0,1}

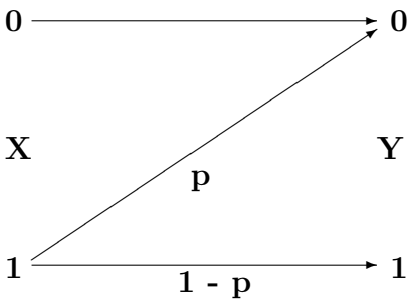
Симметричность: $p(0 \to 1) = p(1 \to 0)$



Другие модели каналов

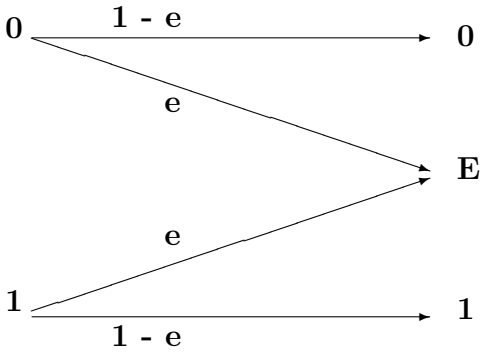
Z-канал (оптический)

0-light on, **1**-light off



MAC-канал (стирающий)

$p(X = 0) = p_0$.



BER – bit error rate – это средняя вероятность ошибки одного бита передаваемой информации.

1. Мобильные каналы: $BER \approx 10^{-2}$.
2. Проводные каналы: $BER \approx 10^{-5}$.
3. Оптоволоконные каналы: $BER \approx 10^{-12}$.

При помехоустойчивом кодировании возможны следующие две стратегии:

- Исправление ошибки за счет избыточности (FEC – forward error correction).
- Обнаружение ошибок с последующим запросом на повторную передачу ошибочно принятой информации (ARR – automatic repeat request).

Каждая из этих стратегий имеет свои достоинства и недостатки. Первая стратегия требует значительного увеличения избыточности, однако при сокращает расходы на исправление ошибок. Вторая уменьшает общую избыточность, однако увеличивает накладные расходы при повторной пересылке пакета. Поэтому, вторая стратегия выгоднее в тех случаях, когда величина BER является небольшой.

Области применения помехоустойчивого кодирования

1. Хранение информации с высокой плотностью записи – CD-ROM, DVD.
2. Передача данных при ограниченной мощности сигнала –спутниковая и мобильная связь.

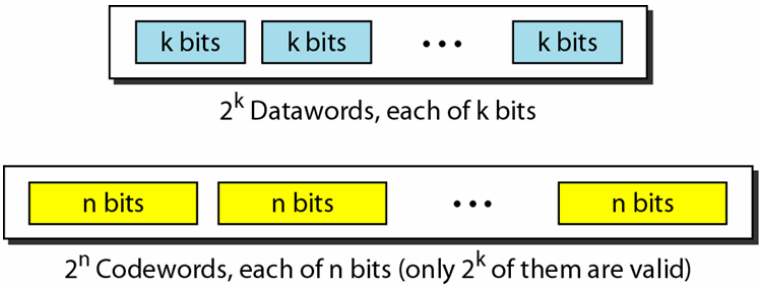


Рис. 3.5: Блоковые коды

3. Передача информации по сильно зашумленным каналам – высокоскоростные проводные линии связи, мобильная связь.

4. Передача данных по каналам связи с повышенными требованиями к надежности информации – вычислительные сети, линии передачи со сжатием.

Пример кодирования.

Информационное слово	Кодовое слово
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

В общем случае, кодирование – это отображение из множества информационных слов I в множество кодовых слов C:

$$f: \ I \subseteq B^k \ \longrightarrow C \subseteq B^n, \quad B = \{b_1, \, b_2 \, \ldots, b_n\}.$$

В качестве алфавита B обычно берется множество $B = \{0, 1\}$.

Расстояние Хэмминга

На множестве кодовых слов вводится некоторая метрика, называемая *расстоянием Хэмминга.* Расстояние Хэмминга между двумя словами есть число разрядов, в которых эти слова различаются.

Пример. Расстояние Хэмминга $d(000, 011) = 2$.

Декодирование – исправление ошибки, если она произошла. Декодирование неправильного слова выполняется в сторону правильного слова, до которого расстояние *наименьшее*.

Пример. Множество кодовых слов 00000, 01101, 10110, 11011. Если полученное слово 10000, то декодируем в «ближайшее» слово 00000. Если же полученное слово равно 11000, то можно установить только факт ошибки, т.к. существует два варианта восстановления: 11000 может перейти в 00000 или в 11011.

Выводы: Если в процессе передачи по зашумленному каналу кодовое слово отобразится в другое кодовое слово, не совпадающее с переданным, то происходит *необнаруживаемая* ошибка –ошибка декодирования.

Хорошие коды должны иметь такую структуру, чтобы была возможность не только обнаруживать, но и исправлять ошибки.

3.5 Теория кодирования и криптография

Всякая технология не совершенна, как и любой другой аспект реального мира. При передаче информации по каналу связи, не вся переданная информация может быть получена без изменения сообщения; некоторые данные могут быть потеряны или изменены при передаче. Одно переключение цифры может привести к серьезному недопониманию или сбою связи. Успешная передача информации требует защиты от возможных случайных ошибок, возникающих в канале. Математические свойства конечных полей позволяют использовать для решения этой задачи коды с исправлением ошибок. Хотя такие коды могут показаться абстрактной концепцией, результаты широко применяются в реальных приложениях. Например, достаточно части штрих-кода для его правильного сканирования или немного поврежденные QR-коды могут передавать информацию. Коды с исправлением ошибок проектируются таким образом, чтобы гарантировать восстановление исходного сообщения при наличии определенного количества точных данных, несмотря на прогнозируемое количество отсутствующих данных или ошибок. По мере того, как количество исправляемых ошибок с помощью кода увеличивается, скорость передачи уменьшается. Поэтому, кодировщик должен учитывать содержание сообщений и определять баланс между емкостью кода и эффективностью.

Кодирование - это частный случай преобразования информации. В зависимости от целей различают следующие виды кодирования:

1. Эффективное (оптимальное) кодирование – кодирование, выполняемое с целью улучшения представления информации (например, ее сжатие).

2. Помехоустойчивое кодирование – кодирование, позволяющее и/или восстанавливать информацию, испорченную при передаче ее по каналу связи.

3. Криптографическое кодирование – кодирование с засекречиванием информации.

Отцом современной теории кодирования называют Клода Элвуда Шеннон (Shannon) (1916 — 2001) — американского инженера и математика. В своих работах 1948-49 годов он определил *количество информации* через энтропию — величину, известную в термодинамике и статистической физике как мера неупорядоченности системы, а за единицу информации принял то, что впоследствии окрестили "битом", то есть выбором одного из двух равновероятных вариантов.

3.6 Линейные коды

В области математики и теории информации линейный код — это важный тип блокового кода, использующийся в схемах определения и коррекции ошибок. Линейные коды, по сравнению с другими кодами, позволяют реализовывать более эффективные алгоритмы кодирования и декодирования информации. В процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Контроль целостности данных и исправление ошибок — важные задачи на многих уровнях работы с информацией (в частности, физическом, канальном, транспортном уровнях модели OSI). В системах связи возможны несколько стратегий борьбы с ошибками:

- обнаружение ошибок в блоках данных и автоматический запрос повторной передачи поврежденных блоков — этот подход применяется в основном на канальном и транспортном уровнях;

- обнаружение ошибок в блоках данных и отбрасывание поврежденных блоков — такой подход иногда применяется в системах потокового мультимедиа, где важна задержка передачи и нет времени на повторную передачу;

- исправление ошибок (англ. forwarderrorcorrection) применяется на физическом уровне.

Параметры линейного кода

Помехоустойчивое кодирование сообщений дискретного источника информации заключается в том, что поступающие k-символьные информационные комбинации $A = (a_1, a_2, \ldots, a_k)$ дополняются $n - k$ избыточными символами до n-символьных кодовых комбинаций $S = (s_1, s_2, \ldots, s_n)$. В процессе передачи последних по каналу связи под действием помех отдельные символы кодовой комбинации искажаются и трансформируются на приемной стороне в другие символы из используемого для передачи алфавита.

Наиболее употребимы двоичные линейные коды. Такой код определяется как множество из 2^k кодовых n-символьных комбинаций, образующих линейное подпространство размерности k. Линейные коды обозначаются (n, k, d_0). Здесь

n – длина кода, число символов в кодовых словах или размерность пространства кодовых комбинаций;

k – число информационных символов или размерность кода;

$n - k$ – количество проверочных или избыточных символов.

Числа n и k определяют относительную скорость передачи информации кодом, равную k/n двоичных единиц на 1 символ кодовой комбинации.

Третий параметр линейного кода – кодовое расстояние d_0 характеризует корректирующую способность помехоустойчивого кода и вводится как минимальное из расстояний Хэмминга при попарном сравнении кодовых слов. С кодовым расстоянием связаны кратности обнаруживаемых q_f и исправляемых q_{cor} ошибок, произошедших в пределах одной кодовой комбинации:

$$d_0 > q_f + 1 \quad \text{или} \quad d0 \geq 2q_{cor} + 1.$$

Число q_f указывает, что код способен обнаруживать все конфигурации вектора ошибки, вес которых $q \leq q_f$. Число q_{cor} указывает, что код способен исправлять все конфигурации вектора ошибки, вес которых $q \leq q_{cor}$.

При совмещении процедур обнаружения и исправления ошибок, причем $q_f > q_{cor}$ соотношение между d_0, q_f и q_{cor} имеет вид:

$$d_0 > q_f + q_{cor} + 1$$

При фиксированных n и k большей помехоустойчивостью обладают коды с большим кодовым расстоянием.

3.7 Коды обнаружения и исправления ошибок

Корректирующие коды — коды, служащие для обнаружения или исправления ошибок, возникающих при передаче информации под влиянием помех, а также при ее хранении.

Для этого при записи (передаче) в полезные данные добавляют специальным образом структурированную избыточную информацию, а при чтении (приеме) ее используют для того, чтобы обнаружить или исправить ошибки. Естественно, что число ошибок, которое можно исправить, ограничено и зависит от конкретного применяемого кода.

С кодами, исправляющими ошибки, тесно связаны коды обнаружения ошибок. В отличие от первых, последние могут только установить факт наличия ошибки в переданных данных, но не исправить ее. В действительности, используемые коды обнаружения ошибок принадлежат к тем же классам кодов, что и коды, исправляющие ошибки. Фактически, любой код, исправляющий ошибки, может быть также использован для обнаружения ошибок (при этом он будет способен обнаружить большее число ошибок, чем был способен исправить).

По способу работы с данными коды, исправляющие ошибки делятся на блоковые, делящие информацию на фрагменты постоянной длины и обрабатывающие каждый из них в отдельности, и сверхточные, работающие с данными как с непрерывным потоком.

3.8 Блоковые коды

Пусть кодируемая информация делится на фрагменты длиной k бит, которые преобразуются в кодовые слова длиной n бит. Тогда соответствующий блоковый код обычно обозначают (n, k). При этом число $R = k/n$ называется скоростью кода.

Если исходные k бит код оставляет неизменными, и $n-k$ добавляет проверочных, такой код называется *систематическим*, иначе *несистематическим*. Задать блоковый код можно по-разному, в том числе таблицей, где каждой совокупности из k информационных бит сопоставляется n бит кодового слова.

Однако, хороший код должен удовлетворять, как минимум, следующим критериям:

- способность исправлять как можно большее число ошибок,

-как можно меньшая избыточность,

-простота кодирования и декодирования.

Нетрудно видеть, что приведенные требования противоречат друг другу. Именно поэтому существует большое количество кодов, каждый из которых пригоден для своего круга задач. Практически все используемые коды являются линейными. Это связано с тем, что нелинейные коды значительно сложнее исследовать, и для них трудно обеспечить приемлемую легкость кодирования и декодирования. Расскажем, поподробнее, о данном типе кода.

Блочный код - в информатике тип канального кодирования. Он увеличивает избыточность сообщения так, чтобы в приёмнике можно было расшифровать его с минимальной (теоретически нулевой) погрешностью, при условии, что скорость передачи информации (количество передаваемой информации в битах в секунду) не превысила бы канальную производительность.

Главная характеристика блочного кода состоит в том, что это – канальный код фиксированной длины (в отличие от такой схемы кодирования источника данных, как кодирование Хаффмана, и в отличие таких методов канального кодирования, как конволюционное кодирование («сверточное» кодирование)). Обычно, система блочного кодирования получает на входе k-значное кодовое слово W, и преобразовывает его в n-значное кодовое слово $C(W)$. Это кодовое слово и называется блоком. Блочное кодирование было главным типом кодирования, используемого в ранних системах мобильной коммуникации.

Формальное определение: Блочный код - код, кодирующий последовательности из набора символов алфавита S в кодовые слова, преобразуя каждый символ из S отдельно. Пусть $\{k_1, k_2, \ldots, k_m\}$- последовательность натуральных чисел, каждое меньше |S|. Если $S = \{s_1, s_2, \ldots, s_n\}$ и некоторое слово W из алфавита S записано как $W = \{s_{k_1}, s_{k_2}, \ldots, s_{k_m}\}$, тогда кодовым словом, соответствующим W, а именно, C(W), будет:

$$C(W) = \{C(s_{k_1}),\ C(s_{k_2}), \ldots,\ C(s_{k_m})\}$$

Компромисс между эффективностью (большей скоростью передачи информации) и способностями исправления может также быть виден

при попытке задать фиксированную длину ключевого слова, и фиксированную возможность исправления (представленную расстоянием Хемминга d) и максимизируют общее количество ключевых слов.

$[n, d]$ — максимальное число ключевых слов для данной длины ключевого слова n и расстояния Хемминга d.

Когда C - двойной блочный код, состоящий из ключевых слов длиной n бит, тогда информационная норма C определяется как $\log_2 A/n$.

В случае, когда первые k бит ключевого слова - независимые информационные биты, то информационная норма будет иметь вид:

$$\log_2 2^k/n = \frac{k}{n}$$

3.9 Идея кодирования двоичных сообщений

Двоичные коды для исправления ошибок могут определить, была ли изменена цифра и в каком месте. Поскольку единственными возможными цифрами в двоичных кодах являются 1 и 0, обнаруженные ошибочные цифры можно просто переключить на противоположное значение. Таким образом, даже несмотря на то, что коды называются «исправляющими ошибки», двоичные коды должны только определять местонахождение ошибок. Чтобы исправить возможные ошибки, в код добавляется избыточность, так что измененные сообщения продолжают напоминать исходные сообщения в достаточной степени для их точной интерпретации. Подобно тому, как человеческий мозг может рассуждать, используя слова с ошибками, добавление избыточности позволяет компьютерам анализировать ошибочную информацию. Для простых двоичных кодов с исправлением ошибок избыточность добавляется в виде контрольных бит. Биты проверки четности - это дополнительные цифры, добавляемые к сообщению вместе с желаемыми данными, которые предоставляют компьютерам информацию, необходимую им для выявления «орфографических ошибок».

Контрольные цифры не добавляются случайным образом. Они вычисляются, используя матричные или полиномиальные операции. В общем, сообщение с общим количеством бит n из которых r являются контрольными, содержит $k = n - r$ информационных разрядов. Таким образом, результирующий двоичный код имеет 2^k кодовых слов длины n, так как всего имеются 2^k информационных слов которые могут быть успешно

переданы. Кодировщик может управлять количеством возможных кодовых слов, которые можно сгенерировать и изменить скорость R передачи информации, где $R = k/n$ регулируя эту величину. При увеличении избыточности за счет добавления контрольных цифр, увеличивается надежность обнаружения ошибок, но также снижается скорость передачи информации $R = k/n = (n - r)/n$. Например, если кодировщику требуется возможность отправлять 512 уникальных блоков информации, для их кода потребуется $512 = 2^9$ уникальных кодовых слов. Таким образом, сообщения будут иметь длину девять бит. Предположим, что тот же кодировщик решает добавить три контрольных цифры, чтобы вместо этого иметь сообщения длиной $n = 9 + 3 = 12$, тогда из возможных 2^{12} сообщений, которые можно построить, 2^9 будут кодовыми словами, так что только $512/4096 \cdot 100 = 12,5\%$ из возможных сообщений являются кодовыми.

Кодировщики хотели бы, чтобы вероятность того, что ошибки передачи преобразуют одно правильное кодовое слово в другое правильное кодовое слово (в этом случае ошибка не обнаружится), была низкой. В представленном примере надежность кода равна $100 - 12,5 = 87,5\%$ при скорости передачи $R = k/n = 9/12$. Если кодировщик недоволен уровнем надежности, избыточность можно увеличить, добавив больше контрольных цифр; однако они должны согласиться с более низкой скоростью передачи. Если кодировщик решит добавить вместо трех - шесть проверок, то только $512/32768$ или $1,5625\%$ возможных сообщений являются кодовыми словами, и надежность кода есть $100 - 1,5625 = 98,4375\%$ при скорости передачи информации $R = 9/15$.

Если у кодировщика есть основания полагать, что вероятность возникновения ошибки в конкретном канале - это некоторое $e,\ 0 < e < 1$, то как только желаемый баланс между надежностью и скоростью передачи определен, кодировщик может построить соответствующий код для информации.

3.10 Исправление одной ошибки в двоичном сообщении

В 1950 году математик и инженер Ричард Хэмминг опубликовал первую теорию линейных кодов, задаваемых матрицами. Такие линейные коды теперь называются кодами Хэмминга. По определению, код является линейным тогда и только тогда, когда его кодовые слова представляют собой набор векторов C, удовлетворяющих уравнению $HxC^t = 0$ для мат-

рицы H (проверочная матрица). В частности, длина кода равна числу столбцов проверочной матрицы H. Теория Хэмминга послужила толчком для математиков для расширения его результатов. Теория утверждает, что коды Хэмминга обладают способностью исправлять все возможные одиночные ошибки тогда и только тогда, когда все столбцы матрицы H различны и не равны нулю. Соответствующая матрица Хэмминга может быть построена путем введения конечного поля характеристики два и представления столбцов матрицы возрастающими порождающего элемента поля a. Таким образом, первый столбец представляет $a^0 = 1$ или $\{1, 0, 0, 0\}$, а второй столбец представляет $a^1 = a$ или $\{0, 1, 0, 0\}$ и так далее. Предположим, Алиса хочет отправить Бобу сообщение из n бит и ожидает, что произойдет одна ошибка. Сначала ей нужно определить конечное поле порядка n характеристики два и выбрать элемент, генерирующий поле, чтобы построить матрицу, связанную с кодом. Как только матрица установлена, она может определить информационную емкость кода, найдя базис для ядра матрицы H. Результирующая размерность ядра равна количеству информационных разрядов, в которых Алиса может сформировать кодовое слово. Затем ей нужно умножить свое информационное слово на транспонированную матрицу ядра, чтобы добавить контрольные цифры, и, таким образом, правильно сгенерировать полное сообщение. Имея все n разрядов, Алиса может отправить сообщение Бобу. По построению, Боб может пропустить сообщение через матрицу H, связанную с кодом Хэмминга (с помощью умножения матриц), чтобы проверить наличие ошибки. Сначала она вычисляет так называемый синдром, взяв произведение проверочной матрицы Хэмминга и полученного сообщения. Поскольку Алиса разработала кодовое слово как линейную комбинацию базисных векторов ядра (порождающая матрица кода), кодовое слово находится в ядре самой матрицы. Следовательно, если синдром представляет собой нулевой элемент $\{0, 0, 0, 0\}$, то она предполагает, что полученное сообщение правильное. Это тот момент, когда мера надежности кода становится важной. Если ошибка превратила сообщение в другое кодовое слово, то Боб неверно истолкует сообщение Алисы. Если вместо этого синдром отличен от нуля, значит, она знает, что произошла ошибка. В частности, она знает, что ошибка расположена в позиции сообщения, соответствующей столбцу матрицы H кода Хэмминга, совпадающему с синдромом. Это происходит потому, что когда полученное сообщение проходит через матрицу H, только ошибки приведет к ненулевому выходу.

Проиллюстрируем сказанное с помощью кода системы Mathematica. Предположим, что сообщения имеют длину $n = 15$, а конечное поле определяется расширением поля над конечным полем из двух элементов посредством корня неприводимого многочлена $m(x) = 1 + x^3 + x^4$.

Сначала необходимо убедиться, что $m(x)$ является неприводимым многочленом по модулю 2. Все неприводимые в поле размера $2^4 - 1 = 15$ обязательно делит $x^{15} - 1$, и таблица ниже показывает, что $m(x)$ является одним из таких неприводимых многочленов.

Затем определяем выбранное расширение GabField:

$$TableForm[\{Map[First,\ FactorList[x^{15} - 1, Modulus- > 2]]\}]$$

$$1 \quad 1 + x \quad 1 + x + x^2 \quad 1 + x + x^4 \quad 1 + x^3 + x^4 \quad 1 + x + x^2 + x^3 + x^4$$

Затем определяем выбранное расширение GabField:

$$<< FiniteFields'$$

$$GabField = GF[2, \{1, 0, 0, 1, 1\}];$$
$$FieldIrreducible[GabField,\ x]$$
$$1 + x^3 + x^4$$

Обозначим через Minimala минимальный многочлен для элемента a:

$$Minimala = 1 + x^3 + x^4;$$

$$a = GabField[\{0, 1, 0, 0\}];$$
$$TableForm[\{Map[First,\ FactorList[x^{15} - 1, Modulus- > 2]]\}]$$
$$Map[First,\ FactorList[x^{15} - 1, Modulus- > 2]]/.x- > a\}]$$

$$1 \quad 1 + x \quad 1 + x + x^2 \quad 1 + x + x^4 \quad 1 + x^3 + x^4 \quad 1 + x + x^2 + x^3 + x^4$$

$$\{1, 1, 0, 0\}_2 \quad \{1, 1, 1, 0\}_2 \quad \{0, 1, 0, 1\}_2 \quad 0 \quad \{0, 1, 1, 0\}_2$$

Таким образом, убеждаемся, что элемент a можно взять в качестве порождающего элемента поля. Выпишем степени элемента a (элементы

поля):

$$\begin{pmatrix} 0 \\ \{0,1,0,0\}_2 \\ \{0,0,1,0\}_2 \\ \{0,0,0,1\}_2 \\ \{1,0,0,1\}_2 \\ \{1,1,0,1\}_2 \\ \{1,1,1,1\}_2 \\ \{1,1,1,0\}_2 \\ \{0,1,1,1\}_2 \\ \{1,0,1,0\}_2 \\ \{0,1,0,1\}_2 \\ \{1,0,1,1\}_2 \\ \{1,1,0,0\}_2 \\ \{0,1,1,0\}_2 \\ \{0,0,1,1\}_2 \end{pmatrix}$$

Построим вручную матрицу Example1 из последовательных элементов поля:

$$\begin{aligned} Example1 = \{&1,0,0,0,1,1,1,1,0,1,0,1,1,0,0\}, \\ &\{0,1,0,0,0,1,1,1,1,0,1,0,1,1,0\}, \\ &\{0,0,1,0,0,0,1,1,1,1,0,1,0,1,1\}, \\ &\{0,0,0,1,1,1,1,0,1,0,1,1,0,0,1\}\}; \end{aligned}$$

Построим проверочную матрице кода Хэмминга SingleHamming:

$$SingleHamming = MatrixForm[Example1]$$

$$\begin{pmatrix} 0 \\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0 \\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \end{pmatrix}$$

Строим порождающую матрицу кода SingleHammingCode:

Допустим, информационные разряды равны $M = 001101100101$, строим кодовое слово Correct, умножая M на порождающую матрицу:

Допустим, произошла ошибка во втором разряде и мы получили слово $Received = 100100101100101$. Вычисляем синдром. В представленном примере синдром соответствует второму столбцу матрицы Хэмминга. Следовательно, ошибка произошла во второй позиции полученного

```
SingleHammingCode = Reverse[Mod[NullSpace[Example1], 2]];
MatrixForm[SingleHammingCode]
```

$$\begin{pmatrix}
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

```
M = {0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1};

Correct = Mod[(Transpose[SingleHammingCode].M), 2]

{1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1}
```

сообщения, как было задумано. Этот метод сопоставления работает, потому что умножение матриц работает с отдельными записями сообщений в последовательном порядке. Таким образом, переключенная цифра в конкретной позиции слова приводит к изменению выходной матрицы каждый раз, когда умножение матриц выполняется в этой позиции в матрице. Обратите внимание на то, что две ошибки в сообщении приведут к сбою декодирования, поскольку сложение двух единиц равно нулю в двоичном формате. Хотя коды Хэмминга такой формы эффективны для исправления одной ошибки, они не могут исправлять любые дополнительные ошибки.

3.11 Эффективность помехоустойчивого кодирования

При проектировании систем передачи информации оценка достоверности обмена информацией определяется вероятностью искажения двоичного символа передаваемого сообщения P_{er}.

Для двоичной последовательности, содержащей n символов, при

безызбыточном кодировании вероятность правильного приема последовательности n равна:

$$P_{pr}(n) = (1 - p_{er})^n,$$

а вероятность ошибки в принятой последовательности

$$P_{er}(n) = 1 - (1 - p_{er})^n.$$

Эту формулу можно записать в следующем виде: ,

$$P_{er}(n) = C_n^1 p_{er}^1 - C_n^2 p_{er}^2 + \ldots \pm C_n^i p_{er}^i,$$

где C_n^i – число сочетаний из n по i.

Использование избыточных кодов позволяет исправлять или обнаруживать в зависимости от кодового расстояния ошибки той или иной кратности. При независимых ошибках вероятность появления кратных ошибок определяется по формуле Бернулли:

$$P_{er}(i, n) = C_n^i p_{er}^i (1 - p_{er})^{n-i},$$

где $i = 1, 2, 3 \ldots$ -– кратность ошибок.

3.12 Исправление двух и более ошибок - код БЧХ

В 1960 году только через десять лет, результаты Хэмминга были обобщены Бозуом, Рой-Чоудхури и Хоккингемом на случай обнаружения более одной ошибки. Названные в их честь коды БЧХ показывают, что, построив простую модификацию исправляющего одиночную ошибку кода Хэмминга, можно успешно исправить две ошибки. Подобно тому, как проверочная матрица Хэмминга для единичной ошибки представляет возрастающие степени порождающего поле элемента a, возрастающие степени a^3 добавляются к соответствующей матрице Хэмминга под строками, выделенными для a. Эта модификация создает матрицу, которая имеет в два раза больше строк, чем проверочная матрица Хэмминга, и , новая матрица дает возможность исправлять вдвое больше ошибок. Важно отметить, что используется элемент a^3, а не a^2, поскольку синдром, соответствующий a^2, является квадратом синдрома, соответствующего матрице Хэмминга в силу того, что основное поле имеет характеристику 2. Так же, как и в случае однократного исправления ошибок, кодовые слова предназначены для включения в ядро кодовой матрицы. Однако ненулевой синдром полученного сообщения теперь состоит из двух частей, первой и второй половины цифр, S_1 и S_3 соответственно. Элвин Берлекамп

предложил использовать квадратное уравнение из вычисленных частей синдрома S_1 и S_3, так чтобы корни уравнения давали местоположения ошибок. Хотя Берлекамп считается создателем эффективного алгоритма и формул для исправления более, чем одной ошибки, возможность исправления многократных ошибок было доказано теоретически Боузом и Рой-Чоудхури в 1960 году. Поскольку вычисления выполняются в базовом поле из двух элементов, возведение в квадрат является линейной операцией.

Таким образом, решения могут быть найдены с помощью серии матричных операций. После вычисления корней их обратные значения соответствуют конкретным столбцам кодовой матрицы, в которых произошла ошибка так же, как и в случае с исправлением одной ошибки. Например, рассмотрим то же расширение поля, что и в предыдущем примере, но теперь предположим, что Боб получает сообщение от Алисы с двумя ошибками вместо одной. Во-первых, Боб вычисляет два синдрома, S_1 и S_3, взяв произведение сообщения и проверочной матрицы и разделив его на две части. Затем она может вставить их в полином Берлекампа для обнаружения ошибок. Выполняя соответствующие линейные вычисления, необходимые для нахождения корней многочлена, вычисляя обратные значения выходных данных и сравнивая результат с матрицей Хэмминга, чтобы найти совпадающие столбцы, ошибки могут быть успешно обнаружены.

Как и раньше, позиции совпадающего столбца сигнализируют о позициях сообщения, в котором произошли ошибки. Код Mathematica для вычислений приводится ниже. Проверочная матрица для исправления двойной ошибки строится как представления последовательных степеней элементов a и a^3.

Обозначим через DoubleHamming проверочную матрицу

Верхние четыре строки задают последовательные степени a, а нижние четыре строки - последовательные степени a^3. Строим порождающую матрицу:

Берем некоторое информационное слово M и, умножив порождающую матрицу на столбец M, получим кодовое слово Correct:

Допустим, в канале произошло две ошибки и Боб получил искаженное слово Received:

Теперь выпишем неприводимые многочлены над полем, порожденным двоичным многочленом $1 + x^3 + x^4$:

```
α = GabField[{0, 1, 0, 0}]
MatrixForm[Table[{α^i, (α^i)^3}, {i, 0, 14}]]
```

$\{0, 1, 0, 0\}_2$

1	1
$\{0, 1, 0, 0\}_2$	$\{0, 0, 0, 1\}_2$
$\{0, 0, 1, 0\}_2$	$\{1, 1, 1, 1\}_2$
$\{0, 0, 0, 1\}_2$	$\{1, 0, 1, 0\}_2$
$\{1, 0, 0, 1\}_2$	$\{1, 1, 0, 0\}_2$
$\{1, 1, 0, 1\}_2$	$\{1, 0, 0, 0\}_2$
$\{1, 1, 1, 1\}_2$	$\{0, 0, 0, 1\}_2$
$\{1, 1, 1, 0\}_2$	$\{1, 1, 1, 1\}_2$
$\{0, 1, 1, 1\}_2$	$\{1, 0, 1, 0\}_2$
$\{1, 0, 1, 0\}_2$	$\{1, 1, 0, 0\}_2$
$\{0, 1, 0, 1\}_2$	$\{1, 0, 0, 0\}_2$
$\{1, 0, 1, 1\}_2$	$\{0, 0, 0, 1\}_2$
$\{1, 1, 0, 0\}_2$	$\{1, 1, 1, 1\}_2$
$\{0, 1, 1, 0\}_2$	$\{1, 0, 1, 0\}_2$
$\{0, 0, 1, 1\}_2$	$\{1, 1, 0, 0\}_2$

Рис. 3.6: Проверочная матрица

```
DoubleHammingCode = Reverse[Mod[NullSpace[DoubleHamming], 2]];
```

Рис. 3.7: Код Математики для построения проверочного слова

Минимальный многочлен для a^3 обозначим Minimala3:

$Minimala3 = x^4 + x^3 + x^2 + x + 1;$

Берем полиномиальное представление принятого слова Received и находим остатки - синдромы:

Заметим, что полиномы, определенные S_1 и S_3, представляют те же синдромы, что и в смысле матричных операций, а именно $\{1, 0, 0, 0\}$ и $\{0, 1, 0, 1\}$ соответственно.

```
MatrixForm[DoubleHammingCode]
```

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
MatrixForm[Transpose[DoubleHammingCode]]
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Рис. 3.8: Generating Matrix

```
M = {1, 1, 0, 0, 1, 0, 0};
Correct = Mod[Transpose[DoubleHammingCode].M, 2]
{0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0}
```

Рис. 3.9: Слово Correct

```
Correct = {0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0};
Received = {0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0};
```

```
TableForm[{Map[First, FactorList[x^15 - 1, Modulus → 2]],
  Map[First, FactorList[x^15 - 1, Modulus → 2]] /. x → α^3}]
```

1	$1 + x$	$1 + x + x^2$	$1 + x + x^4$	$1 + x^3 + x^4$	$1 + x + x^2 + x^3 + x^4$
1	$\{1, 0, 0, 1\}_2$	$\{0, 1, 1, 0\}_2$	$\{0, 1, 0, 1\}_2$	$\{1, 1, 1, 0\}_2$	0

Рис. 3.10: Irreducible Polynomials

```
ReceivedPoly = x + x^2 + x^3 + x^4 + x^6 + x^7 + x^8 + x^9 + x^12 + x^13;
PolynomialQuotientRemainder[ReceivedPoly, Minimalα, x, Modulus → 2]
PolynomialQuotientRemainder[ReceivedPoly, Minimalα3, x, Modulus → 2]
```

$\{1 + x + x^2 + x^4 + x^9, 1\}$

$\{x + x^2 + x^4 + x^5 + x^7 + x^9, x^2 + x^3\}$

```
(α^3)^2 + (α^3)^3
```

$\{0, 1, 0, 1\}_2$

```
S1 = 1;
S3 = x + x^3;
```

Глава 4

Сведения из теории чисел

Пусть $\mathbf{Z}$ обозначает множество целых чисел. Все рассматриваемые в нашем учебнике числа, если не указано особо, принадлежат $\mathbf{Z}$.

Определение 4.0.1 *Говорят, что два целых числа a и b* сравнимы *по модулю p, записывается,*

$$a \equiv b \,(\bmod p),$$

если $p|(a-b)$ (разность $a-b$ делится на p без остатка).

Отношение сравнения по модулю натурального числа обладает следующими свойствами:

1. Рефлексивность: $a \equiv a \,(\bmod p)$.
2. Симметричность: $a \equiv b \,(\bmod p) \rightarrow b \equiv a \,(\bmod p)$.
3. Транзитивность: $a \equiv b \,(\bmod p) \,\&\, b \equiv c \,(\bmod p) \rightarrow a \equiv c \,(\bmod p)$.

Значит отношение сравнения по модулю является отношением эквивалентности на множестве целых чисел. Классы эквивалентности, образованные целыми числами по этому отношению, называются *вычетами*. Вычет, содержащий число k, обозначается $\overline{k}$. Множество классов вычетов по модулю числа натурального $n > 0$ содержит ровно n элементов, записываемых как $\mathbf{Z}_n = \{\overline{0}, \overline{1}, \ldots \overline{n-1}\}$.

Над вычетами можно выполнять арифметические операции сложения, вычитания, умножения и возведения в степень, а если число n простое или является некоторой степенью простого числа, то и деление. Будем обозначать множество вычетов по модулю n через $\mathbf{Z}_n$.

Отметим, что для любых a и b выполняется формула $\overline{a} + \overline{b} = \overline{a+b}$ (то же для других перечисленных выше операции), поэтому операции над вычетами выполняются как над обычными числами, приводя результат к значению, принадлежащему интервалу $[0, n-1]$ путем вы-

полнения операции вычисления остатка от деления результата на число n (т.е. операции, обозначаемой $\bmod n$). Например, в множестве $\mathbf{Z}_7$ $\overline{2} \cdot \overline{5} = \overline{10} = \overline{3}\,(\bmod 7)$. Чаще пишут просто: $2 \cdot 5 = 3\,(\bmod 7)$.

Множество классов вычетов по модулю n образует структуру, являющуюся *кольцом*. Кольцом K называется непустое множество элементов, на котором определены две арифметические операции *сложения* $+$ и *умножения* $\cdot$, относительно которых выполняются следующие формулы:

1. Ассоциативность по сложению: $(\forall a,\, b,\, c \in K)\ a + (b + c) = (a + b) + c$,
2. Существование нулевого элемента: $(\exists \mathbf{0} \in K)(\forall a \in K)\ a + \mathbf{0} = \mathbf{0} + a = a$,
3. Существование обратного элемента: $(\forall a \in K)(\exists b \in K)\ a + b = b + a = \mathbf{0}$,
4. Ассоциативность по умножению: $(\forall a,\, b,\, c \in K)\, a \cdot (b \cdot c) = (a \cdot b) \cdot c$,
5. Дистрибутивность: $(\forall a,\, b,\, c \in K)\ a \cdot (b + c) = a \cdot b + a \cdot c$,
$(b + c) \cdot a = b \cdot a + c \cdot a$.

Обратный по сложению к a элемент обозначается через $(-a)$. Множество элементов, удовлетворяющих только первым трем свойствам, называется *группой*. Если в группе $< G, + >$ выполняется свойство коммутативности $a + b = b + a$, то группа называется *коммутативной* или *абелевой*. Очевидно, что группа по сложению кольца $\mathbf{Z}_n$ является абелевой группой.

Если модуль n является простым числом, то множество ненулевых элементов кольца $\mathbf{Z}_n$ (обозначаемое через $\mathbf{Z}_n^*$) образует коммутативную группу по умножению, т.е. существует нейтральный элемент $\mathbf{1}$ $a \cdot \mathbf{1} = \mathbf{1} \cdot a$, и для каждого элемента a имеется обратный по умножению a^{-1} со свойством $a \cdot a^{-1} = \mathbf{1}$.

Алгебраические структуры, содержащие абелеву группы по сложению и группу по умножению, связанные законами дистрибутивности, называются *полями*. Конечные поля называют также полями Галуа по имени гениального французского математика Эвариста Галуа (1811 – 1832), исследовавшего эти поля, и обозначают $GF(q)$. Более подробные сведения о конечных полях читатель может получить из монографии Р.Лидла и Г.Нидеррайтера «Конечные поля» [68].

Пусть G–произвольная группа по умножению.

Определение 4.0.2 *Порядком элемента a группы G (обозначается через $ord_G(a)$) называется* наименьшее *число k такое, что $a^k = 1$. Порядком группы называется число ее элементов.*

Следующее свойство, связывающее порядки элементов с порядком группы, широко используется в различных алгоритмах, описанных ниже. Эта теорема была доказана знаменитым французским математиком Жозефом Луи Ланранжем (1736–1813).

Теорема 4.0.1 (Лагранж). *Порядок любого элемента конечной группы является делителем порядка группы.*

Доказательство. Пусть элемент a конечной группы $< G, \cdot >$ имеет порядок $k > 1$. Тогда элементы $a,\ a^2,\ \ldots\ ,\ a^{k-1},\ a^k = 1$ различны и сами образуют группу A, содержащую k элементов и являющуюся подгруппой G. Различные смежные классы $b \cdot A$ для $b \in G$ имеют также мощность k, а объединение их дает в совокупности группу G. Значит, число элементов G равно $k \cdot m$, где m–число смежных классов, откуда вытекает утверждение теоремы.

Пример. Рассмотрим кольцо $\mathbf{Z}_p$ при $p = 29$. Ненулевые элементы этого кольца образуют группу по умножению, порядок которой равен $p-1 = 28$. По теореме Лагранжа порядок любого элемента a этой группы является делителем 28, т.е. может принимать одно из следующих значений: 1, 2, 4, 7, 14 и 28.

Элемент $a \in G$ называется *примитивным* элементом или *генератором* группы, если его порядок $ord_G(a)$ равен порядку группы. Не любая группа имеет генератор. Группа, в которой есть генератор, порождается одним элементом и называется *циклической*.

Малая теорема Ферма

Знаменитый французский математик Пьер Ферма (1601–1665) доказал теорему, которая известна как *малая теорема Ферма*.

Теорема 4.0.2 (Малая теорема Ферма) *Если число p–простое, то для любого натурального числа, не сравнимого с p выполняется сравнение*

$$a^{p-1} \equiv 1 \,(\bmod\ p) \tag{4.0.1}$$

Эта теорема является частным случаем теоремы Лагранжа (теор.4.0.1). Действительно, при простом p множество ненулевых элементов кольца Z_p образует группу по умножению, имеющую $p-1$ элемент. Будем обозначать это множество через Z_p^*. По теореме Лагранжа порядок любого элемента $a \in Z_p^*$ является делителем порядка $p-1$, откуда $a^{p-1} \equiv 1 \,(\bmod\ p)$.

Из теоремы Ферма сразу следует, что если для некоторого $a < p$ выполнено условие $a^{p-1} \not\equiv 1 \,(\mathrm{mod}\ p)$, тогда число p является составным. Однако обращение малой теоремы Ферма не верно – существуют составные числа p, для которых выполняется условие Ферма для каждого a, не сравнимого с p. Такие числа называются числами Кармайкла .

4.1 Функция Эйлера $\varphi(n)$

Знаменитый математик Леонард Эйлер (1707–1783), проживший в России большую часть своей жизни и написавший огромное количество математических трудов в разных областях математики, ввел в обиход функцию φ (Euler' totient function), определенную на целых положительных числах, значением которой на аргументе n является количество положительных чисел, меньших n и взаимно-простых с n.

Очевидно, что для всех $n > 1$ $\varphi(n) < n$, и для простого числа p значение $\varphi(n)$ равно $p-1$. Также выполнены и другие формулы, полезные для вычисления функции $\varphi(n)$:

$$\begin{array}{l} \varphi(p) = p - 1 \text{ для всех простых } \ p, \\ \varphi(p^k) = p^k - p^{k-1} \text{ для простых } p \text{ и натуральных } k\ , \\ \varphi(n_1 \cdot n_2) = \varphi(n_1) \cdot \varphi(n_2) \end{array} \qquad (4.1.2)$$

Алгоритм RSA использует несколько вспомогательный алгоритмов, такие как алгоритм генерации простых чисел, алгоритм вычисления обратного элемента по модулю и алгоритм быстрого возведения в степень, которые мы опишем в следующих параграфах.

4.2 Расширенный алгоритм Евклида

Расширенный алгоритм Евклида (РАЕ) используется во многих криптографических и теоретико–числовых алгоритмах. Он состоит из двух частей. В первой части алгоритма для заданных целых чисел A и B вычисляется их наибольший общий делитель (greatest common divisor d) d. Вычисление Н.О.Д. натуральных чисел A и B выполняется по рекуррентной формуле:

$$\text{Н.О.Д.}(A,\ B) = \text{Н.О.Д.}(B,\ A \bmod B), \qquad (4.2.3)$$

где $A \bmod B$ означает операцию вычисления остатка при целочисленном делении A на B. Производится последовательное использование этой

формулы, пока остаток от деления первого операнда на второй не станет равным 0. Последнее ненулевое значение второго операнда и есть искомый общий делитель:

```
int Euclid(int A,  B)
{
while (A mod B !=0) {
   int C=A mod B;
    A=B; B=C ; }
return B;
}
```

Для решения уравнений вида $Ax + By = d$, где A, B – заданные числа, а d – их наибольший общий делитель, используется *расширенный* алгоритм Евклида. Первая часть РАЕ в результате которой мы находим Н.О.Д. d, выполняется также, как описано выше. Значения A, B, а также целую часть и остаток от деления A на B сохраняются в таблице, содержащей 4 столбца. Третий столбец содержит остаток от деления A на B, а четвертый столбец - целую часть от деления A на B.

Во второй части работы алгоритма к таблице добавляются два новых столбца, озаглавленных x и y. Поместим в последнюю строчку столбцов x и y значения 0 и 1. Затем, считая значения x_{i+1} y_{i+1} известными, последовательно вычисляем значения x_i и y_i, $i \geq 0$, по формулам:

$$x_i = y_{i+1}, \qquad y_i = x_{i+1} - y_{i+1} \cdot (A\, div\, B)_i$$

Пример. Решить уравнение $72x + 25y = 1$. Помещаем в первую строчку значения $A = 72$, $B = 25$. Вычисляем $A\, mod\, B$ – остаток от деления A на B, и $[A\,/B]$ – целую часть от деления A на B. Потом переносим значения B и A mod B на строчку вниз и на одну клетку влево. Повторяем вычисления во второй строке. Продолжаем вычисления, пока значение в столбце $A\, mod\, B$ не станет равным 0. Тогда заносим в последнюю строчку столбцов x и y значения 0 и 1, и ведем вычисление снизу вверх по формулам, описанным выше.

A	B	A mod B	[A/B]	x	y
72	25	22	2	8	-25
25	22	3	1	-7	8
22	3	1	7	1	-7
3	1	0	3	0	1

Ответ: Н.О.Д.(72, 25) = 1 – последнее значение в столбце B. Пара $(x, y) = (8, -25)$, дающая решение уравнению $72x + 25y = 1$, берется из первой строки таблицы.

Пример 2. Найти обратный элемент для $e = 7$ по составному модулю $\varphi(n) = 40$ из примера параграфа 5.1.

Решение. Запустим расширенный алгоритм Евклида, взяв $A = \varphi(n) = 40$ и $B = e = 7$. Получим:

A	B	A mod B	[A/B]	x	y
40	7	5	5	3	-17
7	5	2	1	-2	3
5	2	1	2	1	-2
2	1	0	2	0	1

Значение $y = -17$, находящееся в верхней строке, и есть искомое значение обратного элемента:

$$d = y \mod \varphi(n) = -17 \mod 40 = 23$$

Оценка сложности алгоритма Евклида

Расширенный алгоритм Евклида используется во многих криптографических методах, поэтому оценка его производительности играет важную роль в расчетах эффективности криптографических алгоритмов.

Основным фактором в оценке РАЕ является число итераций в главном цикле вычисления новой пары $(A;\ B)$, или, другими словами, число строчек в таблице вычисления вычислений. Чтобы оценить это число, заметим, что на шаге k произвольной итерации возможны два случая:

Случай 1. $B < A/2$. На шаге $k + 1$ новое значение A, равное предыдущему B, будет меньше, чем $A/2$.

Случай 2. $B \geq A/2$. Остаток $r = A \mod B = A - B$ будет меньше $A/2$, и на шаге $k + 2$ новое значение A станет равным остатку $r < A/2$.

В любом случае, после каждой пары итераций первый аргумент A уменьшается более, чем в 2 раза, значит, общее число итераций не может быть больше, чем $2 \log_2 A$. Число операций на каждой итерации постоянно (как при прямом ходе, так и при подъеме при вычислении коэффициентов уравнения $Ax + By = d$), поэтому, оценка РАЕ равна $O(L)$,

где $L = \lceil \log_2 A \rceil$–длина двоичного представления меньшего из чисел.

Эта оценка не является завышенной, т.к. существует последовательность *чисел Фибоначчи*, $\{F_n\}$, на парах соседних элементов которых и достигается эта верхняя оценка. Последовательность или ряд Фибоначчи определяется следующими формулами:

$$F_0 = 1, \quad F_1 = 1, \quad F_{n+2} = F_n + F_{n+1}, \ n \geq 2.$$

Выпишем начальный интервал этого ряда:

$S = \{1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ 233,\ 377,\ 610,\ 987,\ 1597,\ 2584,\ 4181\}$.

4.3 Алгоритм быстрого возведения в степень по модулю

Большинство операций в кольце вычетов $\mathbf{Z}_n$ можно выполнять, выполнив сначала действия с числами, а затем находя остаток от деления результата на модуль n. Однако с операциями возведения в степень и вычисления дискретного (модулярного) логарифма, такой порядок является очень неэффективным. Например, если мы захотим вычислять $2^{199} \pmod{1003}$, используя калькулятор, входящий в состав операционной системы Windows, то результат окажется неверным. В то же время эту же операции несложно выполнить с помощью алгоритма *быстрого возведения в степень по модулю* заданного натурального числа, который мы сейчас опишем.

Предположим, что требуется вычислить $z = a^b \mod n$. Рассмотрим следующий алгоритм:

1. Представим b в двоичный системе исчисления: $b = (b_0\, b_1 \ldots b_k)_2$, $b_i \in \{0,\ 1\}$. Например, $199 = 11000111_2$,

2. Заполним следующую таблицу

b	b_0	b_1	...	b_k
a	a_0	a_1	...	a_k

где $a_0 = a$, $a_{i+1} = \begin{cases} a_i^2 \mod n, \text{ если } b_{i+1} = 0, \\ a_i^2 \cdot a \mod n, \text{ если } b_{i+1} = 1 \end{cases}$ для $i \geq 0$.

Результат появится в последней ячейке второй строки.

Пример. Вычислить $2^{199} \mod 1003$:

b	1	1	0	0	0	1	1	1
c	2	8	64	84	35	444	93	247

Ответ: $2^{199} \mod 1003 = 247$.

Приведем здесь еще один вариант алгоритма быстрого возведения в степень, не требующий предварительного перевода степени в двоичное представление:

```
long powm(long a, long b, long n) {
        long c = 1;
        while (b) {
          if (b%2 == 0)
            { b/ = 2; a = (a * a)% n;}
          else
            { b − −; c = (c * a)% n;}
          }
        return c;
}
```

4.4 Генерация простых чисел. Решето Эратосфена.

Очевидно, что любое простое число, не равное 2, является нечетным. Существуют признаки делимости целых чисел на различные простые числа, например, чтобы число в десятичном виде делилось на 3 и 9 достаточно, чтобы сумма его цифр делилась на 3 и 9 соответственно. Чтобы число делилось на 5, достаточно, что его последняя цифра была 0 или 5.

Такие частные признаки делимости можно использовать, если нужно уменьшить множество кандидатов проверки на простоту или отсечь заведомо составные числа. Альтернативным способом получения простых чисел является *решето Эратосфена*, приписываемое древнегреческому ученому Эратосфену Киренскому, жившему примерно в 276 - 194 г. до н.э.

Для нахождения множества простых до заранее выбранной верхней границы B мы сначала выписываем последовательность всех нечетных чисел от 3 до B. Затем выбираем первое число в списке, т.е. тройку, и оставляя его в списке, вычеркиваем все кратные 3, начиная с 6. Потом переходим ко второму числу списка (пятерке) и вычеркиваем его крат-

ные, оставив саму пятерку и т.д., пока не дойдем до конца списка. В оставшемся списке будут только простые числа.

Следующая теорема дает критерий проверки простоты числа p и одновременно примитивности корня a:

Теорема 4.4.1 *(Критерий примитивности и простоты). Если для некоторых a и p выполнены условия:*

1. $a^{p-1} \equiv 1(\bmod p)$,
2. $a^{(p-1)/q} \not\equiv 1(\bmod p)$ *для* $\forall q|(p-1)$,

тогда число p–простое, и a является примитивным корнем поля GF_p (т.е. генератором группы по умножению поля GF_p).

Пример. $n = 1\ 022\,333\,835\,329\,657, \quad n-1 = 2 \cdot 2957 \cdot 146\,063 \cdot 292\,877.$

$$3^{n-1} \equiv 1(\bmod n),$$
$$3^{(n-1)/2} \equiv -1\,(\bmod n),$$
$$3^{(n-1)/2597} \equiv 324224767363906\,(\bmod n),$$
$$3^{(n-1)/146\,063} \equiv 697302646321792\,(\bmod n),$$
$$3^{(n-1)/292\,877} \equiv 736785752408036\,(\bmod n).$$

Поэтому число n в нашем примере является простым, а 3 является примитивным корнем поля Галуа GF_n.

Отметим, что разбиение $n-1$ в произведение простых сомножителей само является очень сложной задачей, поэтому для длинных чисел этот критерий простоты неприменим.

4.5 Метод пробных делений

Метод пробных делений (the trial division) является наиболее простым методом проверки простоты входного составного числа n или нахождения его делителей. Будем использовать обозначение $\lfloor x \rfloor$ для функции floor(x), равной наибольшему целому числу, не превышающему x (округление вниз). Аналогично, $\lceil x \rceil$ используется для обозначения функции ceil(x), равной наименьшему целому числу, большему или равному x (округление вверх).

Для этого в цикле выполняется пробное деление n на все целые числа от 2 до $\sqrt{n}$:

```
int Tr_div(int n)
{
for(int i = 2; i < ⌊√n⌋; i++)
if (n%i == 0) return i;
return 0}
```

Каждое деление имеет асимптотическую сложность $O(log^2 n)$, поэтому общая сложность метода может быть оценена как $O(n^{1/2} log^2 n)$. Обозначим через L длину двоичного представления числа n, $L = \lceil \log_2 n \rceil$. Тогда можно записать последнюю оценку в более стандартном для теории вычислимости виде:

$$T(n) = O(L^2 \cdot e^{L/2}). \tag{4.5.4}$$

Значит, алгоритм пробных делений имеет экспоненциальную оценку относительно длины входного числа, поэтому этот метод не может быть использован для тестирования больших чисел.

4.6 Решето Аткина

Решето Аткина — быстрый современный алгоритм нахождения всех простых чисел до заданного целого числа.Это оптимизированная версия старинного решета Эратосфена: решето Аткина проделывает некоторую предварительную работу, а затем вычеркивает числа, кратные квадрату простых. Алгоритм был создан А. Аткиным (A. Atkin) и Д. Бернштейном (D. Bernstein) [2].

Ниже представлена упрощенная версия кода, иллюстрирующая основную идею алгоритма – использование квадратичных форм.

```
int limit = 1000;
int sqr_lim;  bool  is_prime[1001];  int x2,  y2;  int i, j; int n;
// Инициализация решета
sqr_lim =  (int) sqrt((long double) limit);
for (i = 0; i <= limit; i++) is_prime[i] = false;
is_prime[2] = true; is_prime[3] = true;
// Предположительно простые - это целые с нечетным числом
// представлений в данных квадратных формах.
// x2 и y2 - это квадраты i и j (оптимизация).
x2 = 0;
for (i = 1;  i <= sqr_lim; i++) {
    x2 + = 2 * i - 1;
    y2 = 0;
    for (j = 1;  j <= sqr_lim;  j++) {
        y2 += 2 * j - 1;
        n = 4 * x2 + y2;
        if ((n <= limit) && (n % 12 == 1 || n % 12 == 5))
```

```
            is_prime[n] = ! is_prime[n];
        // n = 3 * x2 + y2;
        n -= x2; // Оптимизация
        if ((n <= limit) && (n % 12 == 7))
            is_prime[n] = ! is_prime[n];
        // n = 3 * x2 - y2;
        n -= 2 * y2; // Оптимизация
        if ((i > j) && (n <= limit) && (n % 12 == 11))
            is_prime[n] = ! is_prime[n];
        }
    }
// Отсеиваем квадраты простых чисел в интервале [5, √limit].
// (основной этап не может их отсеять)
for (i = 5; i <= sqr_lim; i++) {
        if (is_prime[i]) {
            n = i * i;
        for (j = n; j <= limit; j += n) {
            is_prime[j] = false;
        }
    }
}
// Вывод списка простых чисел в консоль.
printf("2, 3, 5");
for (i = 6; i <= limit; i++) {
// добавлена проверка делимости на 3 и 5. В оригинальной
// версии алгоритма потребности в ней нет.
        if (is_prime[i]) && (i % 3 <> 0) && (i % 5 <> 0){
            printf(" % d " , i); }
}
```

Обоснование алгоритма. Алгоритм основан на следующей теореме Аткина:

Теорема. Пусть n–натуральное число, свободное от квадратов (т.е. не делящееся ни на какой квадрат простого числа) и удовлетворяющее условию $n \equiv 1 \mod 4$. Тогда, n – просто тогда и только тогда, когда

$$\#S = |\{(x, y) : \ x > 0, y > 0, 4x^2 + y^2 = n\}| \text{ – нечетно}$$

Алгоритм полностью игнорирует любые числа, которые делятся на три, пять и семь. Все числа, четные по модулю 60, делятся на два и заведомо не простые. Все числа, равные (по модулю 60) 3, 9, 15, 21, 27, 33, 39, 45, 51 или 57, делятся на три и тоже не являются простыми. Все числа, равные (по модулю 60) 5, 25, 35 или 55, делятся на пять и также не простые. Все эти остатки (по модулю 60) игнорируются.

Все числа, равные (по модулю 60) 1, 13, 17, 29, 37, 41, 49 или 53, имеют остаток от деления на 4 равный 1. Эти числа являются простыми тогда и только тогда, когда количество решений уравнения $4x^2 + y^2 = n$ нечётно и само число не является квадратом (squarefree).

Числа, равные (по модулю 60) 7, 19, 31 или 43, имеют остаток от деления на 6 равный 1. Эти числа являются простыми тогда и только

тогда, когда количество решений уравнения $3x^2 + y^2 = n$ нечётно и само число не является квадратом.

Числа, равные (по модулю 60) 11, 23, 47 или 59, имеют остаток от деления на 12 равный 11. Эти числа являются простыми тогда и только тогда, когда количество решений уравнения $3x^2 - y^2 = n$ нечётно и само число не является квадратом.

Ни одно из рассматриваемых чисел не делится на 2, 3 или 5, значит они не могут делиться и на их квадраты. Поэтому проверка того, что число не является квадратом, не включает чисел 22, 32 и 52.

Оценка сложности решета Аткина

По оценке авторов алгоритм имеет асимптотическую сложность

$$O\left(\frac{n}{\ln \ln n}\right)$$

и требует $O(n^{1/2+o(1)})$ бит памяти. Ранее были известны столь же асимптотически быстрые алгоритмы, но они требовали существенно больше памяти.

4.7 Тест Поклингтона

Если у числа $n-1$ найдено один или несколько простых делителей, то это позволяет ограничить область значений простых делителей числа n или даже показать, что n является простым. Следующая теорема подтверждает это наблюдение:

Теорема 4.7.1 *(H.C. Pocklington). Пусть $n - 1 = F \cdot R$, и полное разложение множителя F на простые множители известно. Тогда, если для некоторого $a < n$ выполняются условия:*

1. $a^{n-1} \equiv 1 \pmod{n}$,
2. *Н.О.Д.*$(a^{(n-1)/q}, n) \neq 1$ *для любого* $q|F$,

тогда любой делитель числа n сравним с 1 по модулю p.

Доказательство. Пусть p–простой делитель числа n. Из п.1 предположений теоремы следует, что порядок k элемента a^R в мультипликативной группе поля GF_p является делителем $(n-1)/F = R$. Из второго

пункта предположений следует, что k не может быть собственным делителем, т.е. $k = F$. Отсюда $F|(p-1)$, т.е. $p = 1 + m \cdot F$ для некоторого целого m.

Следствие. Если $F > \sqrt{n}$, тогда число n–простое.

Действительно, в этом случае, любой нетривиальный делитель p числа n должен быть больше $\sqrt{n}$, что невозможно.

Пример. Пусть $n = 618\,970\,019\,642\,690\,137\,449\,462\,111$. Число $n-1$ имеет полное разложение вида

$$n - 1 = 2 \cdot 3 \cdot 5 \cdot 17 \cdot 23 \cdot 89 \cdot 353 \cdot 397 \cdot 683 \cdot 2113 \cdot 2\,931\,542\,417.$$

Отметим, что наибольший делитель $n-1$, равный $2\,931\,542\,417$, меньше $\lfloor\sqrt{n}\rfloor = 24\,879\,108\,095\,803$.

Базис $a = 2$ не подходит по условию теоремы, т.к. $2^{(n-1)/q} \equiv 1 \pmod{n}$ для всех делителей q. Выполним тест с базой $a = 3$, $p = 2\,931\,542\,417$:

$$m = 3^{(n-1)/p} - 1 \equiv 180\ 591\ 065\ 836\ 317\ 083\ 554\ 066\ 745 \neq \pm 1 \pmod{n},$$

и, Н.О.Д.$(n,\, m) = 1$. Значит, возможные делители числа n имеют вид $1 + k \cdot p < \sqrt{n}$, откуда, $0 < k < 8486$. Простым перебором всех k можно убедиться, что n не имеет простых делителей, и, значит, является простым.

Можно было также вместо выполнения делений применить теорему для F, равного произведению трех наибольших делителей $n-1$ (для них годится та же база $a = 3$), тогда $F > \lfloor\sqrt{n}\rfloor$, откуда сразу следует, что n–простое.

4.8 Генерация простых чисел

Рассмотрим один способ генерации больших простых чисел, основанный на тесте Поклингтона (стр.68) Пусть задано простое число p:

1. Выберем случайным образом чётное число R на промежутке $p \leq R \leq 4p + 2$ и определим $n = pR + 1$.

2. Проверим число n на отсутствие малых простых делителей, разделив его на малые простые числа.

3. Выполним для числа n тест Миллера-Рабина (см.ниже на с.71) с использованием нескольких различных баз $a < p$. Если при одном из

тестов выяснится, что n–составное число, то выберем новое значение R и повторим вычисления.

Оценка эффективности этого метода зависит от плотности распределения простых чисел и расстояния между соседними простыми числами. Вопрос этот является вовсе не простым и зависит от справедливости обобщенной гипотезы Римана (ОГР). Если допустить справедливость ОГР, то этот алгоритм является полиномиальным.

4.9 Символ Лежандра

Определение 4.9.1 *Пусть $n > 1$–целое число. Число a, принадлежащее интервалу $[0,\, n-1]$ называется* квадратичным вычетом *по модулю n, если найдется целое число x такое, что $x^2 \equiv a \pmod n$.*

Если такого x не существует, то a называется *квадратичным невычетом.* Отметим, что ровно половина элементов из интервала $[0,\, n-1]$ является квадратичными вычетами.

Условия того, является ли a квадратичным вычетом по простому модулю p, проверяется с помощью, так называемого, символа Лежандра:

$$\left(\frac{a}{p}\right) = \begin{cases} 1, \text{ если } (\exists\, x)\, x^2 \equiv a \bmod p, \\ -1, \text{ если не}(\exists\, x)\, x^2 \equiv a \bmod p, \\ 0, \text{ если } p \mid a. \end{cases} \tag{4.9.5}$$

Вычисление символа Лежандра может быть выполнено по следующей формуле, полученной Леонардом Эйлером:

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \mod n. \tag{4.9.6}$$

Однако использование этой формулы на практике сопряжено с вычислениями больших степеней, поэтому предпочтительнее пользоваться законом квадратичной взаимности, доказанный Карлом Гауссом в возрасте 17 лет.

Закон квадратичной взаимности: для любых нечетных простых чисел p и q выполняется формула

$$\left(\frac{q}{p}\right) = \left(\frac{p}{q}\right)(-1)^{(p-1)(q-1)/4}.$$

Иначе говоря,

$$\left(\frac{q}{p}\right) = -\left(\frac{p}{q}\right), \text{ если } p \equiv q \equiv 3 \bmod 4, \quad \text{и} \quad \left(\frac{q}{p}\right) = \left(\frac{p}{q}\right), \text{ иначе.}$$

Гаусс в течение своей жизни неоднократно возвращался к этому закону и получил несколько его доказательств, основанных на совершенно различных идеях.

Для быстрого вычисления символа Лежандра полезными являются также следующие формулы:

$$\left(\frac{q}{p}\right) = \left(\frac{q \bmod p}{p}\right), \qquad \left(\frac{q \cdot r}{p}\right) = \left(\frac{q}{p}\right)\cdot\left(\frac{r}{p}\right), \qquad \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}} \mod n.$$

Пример. Вычислить (15/17):

$$\left(\frac{15}{17}\right) = \left(\frac{3}{17}\right)\cdot\left(\frac{5}{17}\right) = \left(\frac{2}{3}\right)\cdot\left(\frac{2}{5}\right) = (-1)\cdot(-1)^3 = 1$$

Для составных чисел n используется символ Якоби, который является обобщением символа Лежандра на произвольные целые числа и обладает следующим свойством:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{r_1} \cdot \left(\frac{a}{p_1}\right)^{r_2} \cdot \ldots \left(\frac{a}{p_k}\right)^{r_k}, \tag{4.9.7}$$

где $n = p_1^{r_1} \cdot p_2^{r_2} \cdot \quad \ldots\ p_k^{r_k}$–разложение n в произведение степеней простых чисел.

4.10 Тест простоты Миллера–Рабина

В качестве критерия проверки, является ли заданное число n простым или составным, может служить следующая теорема:

Теорема 4.10.1 *(Критерий непростоты) Нечетное число $n \geq 3$ является* составным *тогда и только тогда, когда n является либо полным квадратом, либо найдутся два натуральных числа x и y такие, что*

$$x \not\equiv \pm y \,(\bmod n), \ \text{и}\ x^2 \equiv y^2 \,(\bmod n). \tag{4.10.8}$$

Доказательство. Если выполняются условия (4.10.8) , то Н.О.Д.$(n, x^2 - y^2) \neq 1, \neq n$. Обратно, если $n = p\cdot q$, где $p > q$, то определим $x = (p+q)/2$, $y = (p-q)/2$. Очевидно, x и y удовлетворяют (4.10.8).

На критерии непростоты основан известный вероятностный тест Миллера–Рабина, который старается найти пару x и $y = 1$, удовлетворяющие критерию непростоты.

Пусть n–число, которое необходимо проверить на простоту. Представим $n-1$ в виде $n-1 = 2^s \cdot d$, где d–нечетно. Назовем произвольное число $a \in \mathbf{Z}_n^*$ *свидетелем простоты* n, если выполняет одно из следующих условий:

$$1.\ x = a^d \equiv \pm 1 \,(\mathrm{mod}\, n),\ \text{или}$$

$$2.\ (\exists k,\ 0 < k < s)\ x^{2^k} \equiv -1 \,(\mathrm{mod}\, n). \tag{4.10.9}$$

В противном случае, назовем a *свидетелем непростоты* n.

Докажем сначала, что если для числа n найдется хотя–бы один свидетель его непростоты, то n–составное.

Действительно, пусть для некоторого $a \in \mathbf{Z}_n^*$ не выполнен ни один из п.1–2 условий (4.10.9), тогда последовательность

$$x_0 = a^d (\mathrm{mod}\, n),\ x_1 = x_0^2 (\mathrm{mod}\, n),\ \ldots,\ x_{s-1} = x_{s-2}^2 (\mathrm{mod}\, n)$$

не содержит -1. Вычислим x_s, равное $x_{s-1}^2 (\mathrm{mod}\, n)$. Оно не равно 1. Но если n–простое, то $x_s = a^{d \cdot 2^s} = a^{n-1}$ должно равняться по малой теореме Ферма единице. Значит, число n – составное.

Для оценки эффективности теста Миллера–Рабина определим понятие функции Эйлера.

Рабин доказал теорему о том, что если нечетное число $n > 2$–составное, то множество свидетелей его простоты имеет мощность не более $\varphi(n)/4 < n/4$. Отсюда следует, что если при проверке k произвольно выбранных чисел $a < n$ все они окажутся свидетелями простоты n, то n–простое с вероятностью ошибки, не превышающей 4^{-k}. На этом наблюдении строится следующий тест Миллера–Рабина.

Тест Миллера–Рабина

Пусть число $n > 2$–нечетно и $n - 1 = 2^s \cdot d$, где d–нечетно. Для каждого числа a от 2 до $r + 1$, где r – число проверок в тесте, выполним следующие действия:

1. Вычислим $x_0 = a^d (\mathrm{mod}\, n)$.

2. Проверим условие $x_0 \in \{1,\ n-1\}$. Если оно выполнится, тогда a–свидетель простоты. Перейдем к следующему a.

3. Иначе проверим, содержится ли число $n-1$ в последовательность $\{x_1,\ x_2,\ ...,\ x_{s-1}\}$, где каждый последующий x вычисляется по формуле $x_{i+1} = x_i^2 \,(\mathrm{mod}\, n)$.

Если ответ положительный, то a–свидетель простоты. Перейдем к следующему $a \leq r+1$.

Иначе, найден свидетель непростоты n. Завершаем тест с сообщением «число n–составное».

Если после r проверок окажется r свидетелей простоты, то заканчиваем тест с сообщением «n–вероятно простое».

Пример. Пусть $n = 1729$. Разложим $n-1 = 2^6 \cdot 3^3$. Выполним тест Миллера–Рабина для $a = 2$:

$x_0 = 2^{27} \bmod 1729 = 645 \neq 1, \neq n-1,$

$x_1 = x_0^2 \bmod 1729 = 645^2 \bmod 1729 = 1065.$

$x_2 = x_1^2 \bmod 1729 = 1065^2 \bmod 1729 = 1.$

Последующие элементы $\{x_i\}$ для $i = 3, 4, 5$ равны 1, и последовательность $\{x_1,\ x_2,\ ...,\ x_{s-1}\}$, не содержит $n-1$. Значит, 2 является свидетелем непростоты n, и $n = 1729$ – составное число.

Оценка эффективности теста Миллера–Рабина

Следующая лемма была доказана Рабином в предположении справедливости обобщенной гипотезы Римана о распределении простых чисел:

Лемма 4.10.1 *Пусть число n–нечетное число, и* $n-1 = 2^s \cdot d$, *где d–нечетно. Если для всех* x, $0 < x < 2 \cdot (log_2 n)^2$ *выполняется* $x^d \equiv 1 \,(\mathrm{mod}\, n)$, *или* $x^{2^k \cdot d} \equiv -1 \,(\mathrm{mod}\, n)$ *для некоторого* $0 \leq k < s$. *Тогда число n является простым.*

Оценка, приведенная в этой лемме, является полиномиальной, однако, с теоретической точки зрения она не может быть использована, пока не доказана обобщенная гипотеза Римана, которая на сегодняшний день является самой известной из нерешенных «проблем тысячелетия». Кроме того, для практических расчетов эта оценка является сильно завышенной. Вместо нее обычно используется граница порядка $O(log_2 n)$.

4.11 Вероятностный тест простоты Соловея–Штрассена

Тест Соловея — Штрассена опирается на малую теорему Ферма (разд. 4.0.2) и свойства символа Якоби (разд. 4.9):

Теорема 4.11.1 *Если n — нечетное составное число, то количество целых чисел a, взаимно простых с n и меньших n, удовлетворяющих сравнению*

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}, \tag{4.11.10}$$

не превосходит $n/2$.

Алгоритм Соловея — Штрассена

Сначала для алгоритм Соловея — Штрассена выбирается целое число $k \geq 1$. Тест проверки простоты числа n состоит из k отдельных раундов. В каждом раунде выполняются следующие действия:

1. Случайным образом выбирается число $a < n$, и вычисляется $d =$Н.О.Д.(a, n).

2. Если $d > 1$, то выносится решение о том, что n составное. Иначе проверяется сравнение (4.11.10). Если оно не выполнено, то n - составное. Иначе, a является свидетелем простоты числа n.

Если после завершения k раундов найдено k свидетелей простоты, то делаем заключение «n–вероятно простое число».

Вычислительная сложность и эффективность теста

В каждом раунде вероятность отсеять составное число больше $1/2$, поэтому через k раундов тест Соловея–Штрассена определяет простое число с вероятностью ошибки, меньшей 2^{-k}. Поэтому этот тест сравним по эффективности с тестом Ферма, но имеет преимущество перед тестом Ферма в том, что он отсеивает все числа Кармайкла (числами Кармайкла называются нечетные составные натуральные числа n такие, что для каждого натурального a, $1 \leq a < n$, выполнено условие $a^{n-1} \bmod n = 1$).

С другой стороны, он проигрывает тесту Миллера–Рабина, который за k раундов имеет ошибку, меньшую 4^{-k}.

Общая вычислительная сложность алгоритма оценивается как $O(k \log_2 n)$.

4.12 Полиномиальный критерий простоты AKS

Одной из важных проблем, долгое время стоявших перед исследователями, была проблема построения детерминированного алгоритма проверки простоты натуральных чисел, имеющего полиномиальную

оценку времени работы. Алгоритм Миллера–Рабина, упомянутый в предыдущем разделе, имеет полиномиальную оценку, но не является детерминированным. Другие тесты, например тест Поклингтона (разд.4.7), являются детерминированными, но не имеют полиномиальной оценки.

В 2004 г. тремя молодыми индийскими математиками Агравелой, Каялом и Саксеной ([1]) был разработан детерминированный полиномиальный безусловный тест AKS проверки простоты заданного натурального числа. Тест AKS основывается на следующей теореме:

Теорема 4.12.1 *(Agrawel, Kayal, Saxena [2004].) Пусть n–нечетное натуральное число, r–простое число и выполнены условия:*

1. *Число n не делится ни на одно из чисел, меньших или равных r,*
2. *Порядок n в мультипликативной группе $\mathbf{Z}_p^*$ поля GF_p не меньше $(\log_2(n))^2$,*
3. *Для всех a, $0 \le a \le r$, выполнена формула*

$$(X+a)^n \equiv X^n + a \text{ в кольце многочленов } \mathbf{Z}_n[X]/\frac{X^r-1}{X-1}\ . \tag{4.12.11}$$

Тогда число n является простым.

В этой теореме используются вычисления в кольце многочленов $\mathbf{Z}_n[X]$ с коэффициентами, ограниченными сверху числом n, факторизованных по модулю многочлена деления круга

$$\Phi_r(X) = \frac{X^r-1}{X-1} = X^{r-1} + X^{r-2} + \ ... \ + X \ + \ 1.$$

Конечно, если n–просто, то эквивалентность $(X+a)^n \equiv X^n + a \bmod n$ в силу малой теоремы Ферма выполняется и в кольце $\mathbf{Z}_n[X]$, однако эти вычисления слишком громоздки, чтобы их можно было реально выполнить. Суть замечательной идеи Агравелы, Каялы и Саксены состояла в том, чтобы заменить кольцо $\mathbf{Z}_n[X]$ на гораздо меньшее кольцо $\mathbf{Z}_n[X]/\Phi_r(X)$.

На этой теореме основан следующий тест проверки простоты числа n:

1. Проверим, что n не является полным квадратом,
2. Используя числа $r = 2,\ 3,\ 5,\ ...$, найдем наименьшее простое число r такое, что r не является делителем n, и не является делителем $n^i - 1$ для всех $i \in \{0,\ 1,\ 2,\ ...\ , (\log_2 n)^2)\}$.

3. Проверим, что выполнены условия пункта 3 теоремы.

Если эти условия выполнены, то n–простое, иначе n–составное.

Замечание. Несмотря на то, что тест AKS явился решением крупной и долгостоявшей научной проблемы, он является не слишком удобным с практической точки зрения. Проверка условий пункта 3 является настолько громоздкой, что общая оценка времени работы алгоритма достигает $O(\log^{18} n)$ (см. Д. Вентури. Лекции по алгоритмической теории чисел [38]). Поэтому этот тест следует применять лишь в тех случаях, когда надо получить *гарантированное* доказательство того, что число n является простым.

В следующем главе мы выясним, как распределяются простые числа и дадим формулировку знаменитой *проблемы Римана*.

4.13 Извлечение квадратного корня в конечных полях

В реализациях методов квадратичного решета и решета числового поля, описываемых в 4-й и 5-й главах, будет использован алгоритм извлечения квадратного корня в конечных полях, разработанный Шенксом и Тоннелли. Опишем данный алгоритм в этом параграфе.

Рассмотрим конечное поле GF_p, $p > 2$, и элемент a, являющийся квадратичным вычетом по модулю p. Требуется найти x такое, что $a \equiv x^2 \pmod p$.

Представим число $p-1$ в виде $p-1 = 2^r \cdot s$, где s–нечетно. Заметим, что поскольку $p-1$–четно, $r \geq 1$. Пусть z–некоторый квадратичный невычет по модулю p (его можно найти просто перебором по элементам F_p, пока символ Лежандра (z/p) не окажется равным -1).

Рассмотрим 2 случая:
1. $p \equiv 3 \pmod 4$. В этом случае можно сразу найти решение

$$x = a^{\frac{p+1}{4}} \pmod p$$

2. $p \equiv 1 \pmod 4$.

Вычислим $y = z^s \mod p$. Поскольку порядок любого элемента является делителем числа $2^r \cdot s$, то порядок y является делителем 2^r, откуда $y^{2^r} \equiv 1 \pmod p$. Можно также показать, что $y^{2^r-1} \equiv -1 \pmod p$, т.е. порядок элемента y равен в точности 2^r. Вычислим далее элементы

$$\lambda_0 = a^s \pmod p, \quad w_0 = a^{(s+1)/2} \pmod p. \tag{4.13.12}$$

Заметим, что

$$w_0^2 \equiv a \cdot \lambda_0 \,(\bmod p) \quad \text{и} \quad x^2 \equiv a(\bmod p) \to x^{2s} \equiv a^s = \lambda_0(\bmod p). \tag{4.13.13}$$

Поскольку порядок элемента x^s является делителем 2^r, то порядок λ_0 является делителем 2^{r-1}. Идея метода Шенкса–Тоннелли состоит в построении последовательности пар чисел $(\lambda_i,\, w_i)$, удовлетворяющих условию

$$w_i^2 \equiv a \cdot \lambda_i \,(\bmod p), \;\; i = 0,\, 1,\, 2,\, \ldots\,, \tag{4.13.14}$$

причем порядок λ_{i+1} является собственным делителем порядка λ_i, до тех пор, пока порядок очередного λ_i не окажется равным 0. Тогда для найденного i выполнятся условия $\lambda_i = 1$ и

$$w_i^2 \equiv a \,(\bmod p),$$

откуда $x = w_i$ является искомым корнем.

Поскольку исходные значения $(\lambda_0,\, w_0)$, удовлетворяющие (4.13.14), согласно (4.13.13), уже определены, то осталось просто описать формулы для вычисления значений $(\lambda_{i+1},\, w_{i+1})$:

$$\lambda_{i+1} = \lambda_i \cdot y^{2^{r-m}}, \quad w_{i+1} = w_i \cdot y^{2^{r-m-1}}, \tag{4.13.15}$$

где 2^m–порядок элемента λ_i.

Пример. Рассмотрим пример вычисления квадратного корня из $a = 2$ в простом поле GF_p при $p = 41$:

1. Имеем, $p - 1 = 40 = 2^3 \cdot 5$, откуда, $s = 5$, $r = 3$.

2. Вычислим исходные значения $(\lambda_0,\, w_0)$ по формулам (4.13.12):

$$\lambda_0 = a^s \,(\bmod\ p) = 2^5 \,(\bmod\ 41) = 32,$$
$$w_0 = a^{(s+1)/2} \,(\bmod\ p) = 2^3 \,(\bmod\ 41) = 8.$$

3. Найдем порядок элемента λ_0:

$$\lambda_0^2 \bmod\ p = 32^2 \bmod\ 41 = 40 \equiv -1 \,(\bmod\ 41),\ \lambda_0^4 \equiv 1 \bmod\ p.$$

Отсюда, $ord(\lambda_0) = 2^m = 4$, $m = 2$.

4. Будем искать квадратичный невычет. Вычислим символ Лежандра для $z = 3$:

$$\left(\frac{z}{p}\right) = \left(\frac{3}{41}\right) = \left(\frac{41 \bmod 3}{3}\right)(-1)^{(41-1)(3-1)/2} = \left(\frac{2}{3}\right) = -1,$$

значит, $z = 3$ является квадратичным невычетом и может быть использован для вычисления пар $(\lambda_{i+1},\, w_{i+1})$.

5. Найдем $y = z^s (\bmod\ p) = 3^5 (\bmod\ 41) = 38$.

6. Вычислим степень, в которую надо возводить y:
$d = 2^{r-m} = 2^{3-2} = 2, \quad y^d = 3^2 = 9.$

7. Вычислим $\lambda_1 = \lambda_0 \cdot y^d (\bmod p) = 32 \cdot 9 (\bmod 41) = 1$, $w_1 = w_0 \cdot y^{d-1} (\bmod p) = 8 \cdot 3 (\bmod 41) = 24$. Поскольку очередное λ_i оказалось равным 1, то процедура закончена. Корень $x = w_1 = 24$. Выполним проверку:

$$x^2 \bmod\ p = 24^2 \ \bmod\ 41 = 2 = a.$$

4.14 Китайская теорема об остатках

Китайская теорема об остатках позволяет вычислить целое число, если известны его остатки по нескольким простым модулям. Впервые эта теорема была упомянута в трактате китайского математика Сунь Цзы, примерно в третьим веком до н.э.

Теорема 4.14.1 *Если натуральные числа m_1, m_2, ..., m_n попарно взаимно просты, то для любых целых r_1, r_2, ..., r_n таких, что $0 \le r_1 < m_i$ при всех i, найдётся число x, которое при делении на m_i даёт остаток r_i при всех $1 \le i \le n$. Более того, любые два таких числа x_1 и x_2 удовлетворяют уравнению*

$$x_1 \equiv x_2 \ (\bmod m), \ \textit{где}\ m = m_1 \cdot m_2 \cdot \ldots \cdot m_n.$$

В трактате другого китайского математика Джу Шао Квина (Jiushao Qin) (1247 г. н.э.) дается формула для вычисления числа x, удовлетворяющего теореме:

$$x = \sum_{i=0}^{n} r_i \cdot e_i, \ \text{где}\ e_i = \frac{m}{m_i} \cdot \left(\left(\frac{m}{m_i} \right)^{-1} \bmod\ m_i \right), \ 1 \le i \le n. \quad (4.14.16)$$

Заметим, что поскольку число m_i взаимно просто с m/m_i , то обратное число в формуле для e_i всегда существует для $1 \le i \le n$. Кроме того, имеют место равенства

$$\begin{cases} e_i \cdot e_i \equiv e_i \ (\bmod m), \\ e_i \cdot e_j \equiv 0 \ (\bmod m) \text{ при } i \ne j, \end{cases}$$

т.е. компоненты e_i взаимно ортогональны по модулю m.

Алгоритм Гарнера

Для вычисления x может быть использован алгоритм Гарнера (Garner' algorithm), согласно которому x можно вычислить как n-й член последовательности $\{x_i\}$. Последовательности $\{x_i\}$, $\{y_i\}$ строятся по следующим формулам:

$$\begin{cases} y_1 = x_1 = r_1, \\ y_{i+1} = \frac{r_{i+1} - x_i}{m_1 \cdot m_2 \cdot \ldots \cdot m_i} \mod m_{i+1}, \\ x_{i+1} = x_i + y_{i+1} \cdot m_1 \cdot m_2 \cdot \ldots \cdot m_i. \end{cases} \tag{4.14.17}$$

Достоинство этого алгоритма заключается в том, что вычисление каждого последующей пары $(x_{i+1},\, y_{i+1})$ использует только одно предыдущее значение $(x_i,\, y_i)$, что позволяет последовательно уточнять значения корня x.

Пример. Найти наименьшее положительное x, удовлетворяющее системе уравнение:

$$\begin{cases} x \equiv 2\,(\bmod 3\,) \\ x \equiv 5\,(\bmod 7\,) \\ x \equiv 4\,(\bmod 11\,). \end{cases}$$

Решение. В нашем примере $m_1 = 3$, $m_2 = 7$, $m_3 = 11$, $r_1 = 2$, $r_2 = 5$, $r_3 = 4$. Будем вычислять последовательно y_i и x_i, $i = 1,\, 2,\, 3$:

$y_1 = x_1 = 2,$
$y_2 = (r_2 - x_1) \cdot (m_1)^{-1} \mod m_2 = (5-2) \cdot (3)^{-1} \mod 7 = 1$
$x_2 = x_1 + (y_2 \cdot m_1 \mod m_2) = 2 + (1 \cdot 3 \mod 7) = 5,$
$y_3 = (r_3 - x_2) \cdot (m_1 \cdot m_2)^{-1} \bmod m_3 = (4-5) \cdot 21^{-1} \bmod 11 = 1,$
$x_3 = x_2 + y_3 \cdot m_1 \cdot m_2 = 5 + 1 \cdot 3 \cdot 7 = 26.$

Ответ: $x = 26$.

Глава 5

Метод RSA и проблема факторизации

5.1 Алгоритм RSA.

В этом части рассмотрим один из наиболее популярных алгоритмов двухключевой криптографии – метод RSA. Его работы состоит из трех частей:

I. Генерация ключей.

1. Выбираем два произвольных простых числа p и q.
2. Вычисляем их произведение $n = p \cdot q$ и функцию Эйлера
$$\varphi(n) = (p-1) \cdot (q-1)$$
3. Выбираем случайное число e, $2 \leq e < n$, взаимно-простое с e. Последнее означает, что Н.О.Д(n, e)=1. Объявляем число e открытым ключом RSA.
4. Вычисляет элемент d, $1 < d < n$, обратный к e по модулю $\varphi(n)$. Иначе говоря, d должен удовлетворять условию
$$e \cdot d \mod \varphi(n) = 1$$
Для вычисления d необходимо использовать обобщенный алгоритм Евклида (см. раздел "Обобщенный алгоритм Евклида").
Объявляем число d закрытым ключом RSA.

II. Шифрование.

Для шифрования текстовой строки M выполним следующие действия:

1. Разобьем текст на отдельные символы.
2. Заменим последовательность символов последовательностью их кодов (например, в стандартной кодировке Win 1251).

3. Зашифруем последовательность, заменяя каждый код c на шифрокод по формуле:

$$h = enc(c) = c^e \mod n \tag{5.1.1}$$

III. Расшифрование.

Для расшифрования шифростроки $enc(M)$ выполним следующие действия:

1. Расшифруем последовательность, заменяя каждый шифрокод h на код $c = dec(h)$, вычисляемый по формуле:

$$c = h^d \mod n \tag{5.1.2}$$

2. Заменим коды c на символы текста, восстанавливая сообщение.

Корректность операции восстановления исходных символов текста обеспечивается следующей теоремой Эйлера, являющейся обобщением *малой теоремы Ферма*:

Теорема Эйлера. Для любого элемента $a > 0$ кольца вычетов $\mathbf{Z_n}$ по модулю n выполняется следующая формула:

$$a^{\varphi(n)} \mod n = 1$$

Поскольку из условия $e \cdot d \mod \varphi(n) = 1$ следует, что $e \cdot d = k \cdot \varphi(n) + 1$, где $k \in \mathbf{Z}$, то вычисление (5.1.2) даем нам

$$h^d = (c^e)^d = c^{ed} = c^{k \cdot \varphi(n)+1} = c \cdot (c^{\varphi(n)})^k = c \cdot 1^k = c,$$

откуда вытекает справедливость формулы (5.1.2).

Пример. Пусть $p = 11$, $q = 5$.

1. Вычислим $n = p \cdot q = 55$ и функцию Эйлера $\varphi(n) = (p-1) \cdot (q-1) = 10 \cdot 4 = 40$.
2. Возьмем открытый ключ, равным $e = 7$. Проверим условие
$$\text{Н.О.Д.}(\varphi(n), e) \;=\; \text{Н.О.Д.}(40, 7) = 1$$
3. Найдем d из условия $7 \cdot d \mod 40 = 1$. Вычисление d выполнено в примере 2 следующего параграфа. Получим $d = 23$.

   Параметры RSA определены.
4. Зашифруем число $m = 15$:
$$h = enc(15) = m^e \mod n = 15^7 \mod 55 = 5$$
5. Расшифруем шифрокод
$$c = h^d \mod n = 5^{23} \mod 55 = 15$$

5.2 Криптостойкость RSA

Факторизацией целого числа называется его разложение в произведение простых сомножителей. Такое разложение, согласно основной теореме арифметики, всегда существует и является единственным (с точностью до порядка следования множителей). Известно, что задача факторизации является вычислительно сложной задачей. Однако никому пока не удалось получить высоких нижних оценок этого алгоритма. Известно, что эта задача не является также NP-полной.

Криптостойкость RSA базируется на предположении, что не существует быстрых (полиномиальных) алгоритмов факторизации. В силу отсутствия высоких нижних оценок криптостойкость RSA – только гипотеза, подобно *тезису Черча*. Поэтому вопрос о существовании алгоритма факторизации с полиномиальной сложностью на классическом компьютере для выполнения факторизации является одной из важных открытых проблем современной теории чисел. В то же время факторизация с полиномиальной сложностью возможна на квантовом компьютере с помощью алгоритма Шора.

Все методы факторизации в зависимости от их производительности

можно разбить на две группы: экспоненциальные методы и субэкспоненциальные методы. Все эти методы достаточно трудоемки, поэтому требуют значительных вычислительных ресурсов для чисел большой длины. В этой главе мы дадим описание наиболее известных алгоритмов факторизации, имеющих экспоненциальную оценку сходимости.

Среди субэкспоненциальных алгоритмов следует выделить метод *эллиптических кривых* Ленстры, являющийся аналогом $p-1$-метода Полларда, метод *квадратичного решета* Карла Померанса и *метод решета числового поля.* Описание первого из них будет дано в главе "Эллиптические кривые и их использование в криптографии".

5.3 Метод Ферма

Пусть $n = p \cdot q$ – нечетное целое число, являющееся произведением двух неизвестных простых чисел p и q, которые требуется найти. Большинство современных методов факторизации основано на идее, предложенной еще Пьером Ферма, заключающейся в поиске пар натуральных чисел A и B таких, что выполняется соотношение:

$$n = A^2 - B^2. \tag{5.3.3}$$

Алгоритм Ферма может быть описан следующим образом:

1. Вычислим целую часть от квадратного корня из n:

$$x_0 = \lceil \sqrt{n} \rceil.$$

2. Для $x_i = x_0 + i$, $i = 0, 1, 2, \ldots$ будем вычислять значения

$$q(x_i) = x_i^2 - n, \tag{5.3.4}$$

до тех пор, пока очередное значение $q(x_i)$ не окажется равным полному квадрату.

3. Пусть $q(x_i)$ является полным квадратом, например, числа B: $q(x_i) = B^2$. Определим $A = x_i$, откуда из равенства $A^2 - n = B^2$ найдем $n = A^2 - B^2 = (A+B) \cdot (A-B)$, и искомые делители p и q вычисляются, как $p = A + B$, $q = A - B$.

Пример. Пусть факторизуемое число $n = 19\,691$. Вычислим $m = \lceil \sqrt{n} \rceil = 141$. Представим процедуру вычисления делителей n в виде таблицы:

x	q(x)	$\sqrt{q(x)}$
141	190	13,78
142	473	21,75
143	758	27,53
144	1045	32,33
145	1334	36,52
146	1625	40,31
147	1918	43,79
148	2213	47,04
149	2510	50,10
150	2809	53

Из последнего столбца получим: $(140+10)^2 - n = 53^2$, откуда $n = 150^2 - 53^2 = 203 \cdot 97$. Итак, $19\,691 = 203 \cdot 97$, и вычисление потребовало 10 итераций, в каждой из которых было выполнено 1 возведение в степень, 1 вычитание и одно вычисление квадратного корня, т.е. константное число операций.

Оценка производительности метода Ферма

В наихудшем случае, когда q близко к 1, а p близко к n, алгоритм будет работать даже хуже, чем метод пробных делений. Действительно, $A = (p+q)/2$, откуда число итераций в методе Ферма равно

$$Iter(n) = \frac{p+q}{2} - \lfloor n^{1/2} \rfloor \approx \frac{n}{2} - \lfloor n^{1/2} \rfloor,$$

т.е. имеет порядок $0(n)$. Для того, чтобы метод Ферма работал не хуже, чем метод пробных делений необходимо, чтобы $Iter(n)$ было меньше $n^{1/2}$, откуда больший делитель $p < 4n^{1/2}$.

Таким образом, как и метод пробного деления, алгоритм Ферма имеет экспоненциальную оценку и не эффективен для разложения длинных чисел.

Можно улучшить метод Ферма, выполнив сначала пробное деление числа n на числа от 2 до некоторой константы B, исключив тем самым малые делители n до B включительно, и только потом выполнить поиск методом Ферма.

5.4 $(p-1)$–метод Полларда

Этот метод был разработан английским математиком Джоном Полларом в 1974 г. и опубликован в [33].

Пусть n—факторизуемое число, а $1 < p < n$– его простой делитель. Согласно малой теореме Ферма, для любого a, $1 \leq a < p$, выполняется условие $a^{p-1} \equiv 1(\bmod p)$.

Это же сравнение выполнится, если вместо степени $p-1$ взять произвольное натуральное число M кратное $p-1$, т.к. если $M = (p-1) \cdot k$, то $a^M = (a^{p-1})^k \equiv 1^k \equiv 1(\mathrm{mod} p)$. Последнее условие эквивалентно $a^M - 1 = pr$ для некоторого целого r. Отсюда, если p является делителем числа n, тогда p является делителем наибольшего общего делителя Н.О.Д.$(n, a^M - 1)$ и совпадет с Н.О.Д.$(n, a^M - 1)$, если $a^M - 1 < n$. Пусть

$$p - 1 = p_i^{r_1} \cdot p_2^{r_2} \cdot \ ... \ \cdot p_t^{r^t}. \tag{5.4.5}$$

Идея $(p-1)$–метод Полларда состоит в чтобы выбрать M в виде произведения как можно большего числа простых сомножителей или их степеней так, чтобы M делилось на каждый сомножитель $p_i^{r_i}$, входящий в разложение (5.4.5). Тогда, Н.О.Д.$(n, a^M - 1)$ даст искомый делитель. Алгоритм состоит из двух стадий:

Первая стадия (p-1)–алгоритма Полларда

1. Сначала выберем границу B_1.
2. Определим множество P, состоящее из простых чисел и их степеней, меньших границы B_1:

$$P = \{p_1^{r_1},\ p_2^{r_2},\ \ldots p_k^{r_k}\},\ \ p_i^{r_i} < B_1.$$

3. Вычислим произведение

$$M = M(B_1) = \prod_{p_i^{r_i} \in P} p_i^{r_i}$$

4. Выберем произвольное число a, например 2, и вычислим $a^M \mod n$.
5. Вычислим Н.О.Д.$(n, a^M - 1)$, который, если повезет даст искомый делитель числа n.

Пример. Факторизовать $n = 10\,001$. Выберем $B = 10$, тогда, $M(B_1) = 2^3 \cdot 3^2 \cdot 5 \cdot 7 = 2520$. Вычислим, $2^{2520} \mod 10\,001 = 3579$. Найдем, Н.О.Д.$(n, a^M - 1)$ = Н.О.Д.$(10\,001, 3578) = 73$.

Заметим, что при большом значении B_1 число $M(B_1)$ может оказаться чрезвычайно большим (оно сравнимо с $B_1!$). В таких случаях лучше разбить полное произведение $M(B_1)$ на l блоков, состоящих из примерно одинакового числа сомножителей, и вычислив числа M_i как произведение элементов блока i, представить $M(B)$ в виде $M_1 \cdot M_2 \cdot ... \cdot M_l$. Затем, можно вычислить $a^{M(B)}$ как предел последовательности $\{a_i\}$, где $a_1 = a^{M_1} \,(\mathrm{mod} n)$, а последующие a_i вычисляются по формуле:

$$a_{i+1} = a_i^{M_{i+1}} \mod n, \;\; i < l.$$

В этом случае, все вычисления будут выполняться с числами, сравнимыми по модулю с числом n.

Вторая стадия (p-1)–алгоритма Полларда

Если в результате первого этапа алгоритм не выдает требуемого делителя, то можно либо увеличить границу B_1, либо начать вторую стадию работы алгоритма.

Вторая стадия алгоритма предполагает, что существует только один простой множитель q числа $p-1$, значение которого больше границы B_1. Выберем новую границу $B_2 \gg B_1$, например, $B_2 = B_1^2$. Обозначим через b число $a^{M(B)} \mod n$, вычисленное на первой стадии работы алгоритма.

Выпишем последовательность $q_0 < q_1 < ... < q_s$ всех простых чисел на интервале $[B;\, B_2]$. Для построения этого множества можно воспользоваться решетом Эратосфена, либо решетом Аткина (см.гл.1).

Поскольку наличие в последовательности $\{q_i\}$ нескольких составных чисел не испортит работы алгоритма, можно выполнить только частичное просеивание, отсеяв числа, кратные небольшим простым числам. Это ускорит общую работу алгоритма.

Если искомый множитель $p-1$ равен q_i, то для нахождения делителя n, необходимо вычислить $c_i = b^{q_i} \mod n$, и найти Н.О.Д.$(n, {}_i - 1)$. Поскольку, значение q неизвестно, мы должны выполнить последние две операции с каждым числом q_i из интервала $[B_1;\, B_2]$. Полларда предложил следующий вариант организации этой процедуры. Обозначим через δ_i разность между соседними простыми числами $\delta_i = q_{i+1} - q_i$. Возможные значения, принимаемые d_i, лежат в небольшом множестве $D = \{2,\, 4,\, ...,\, 2t\}$. Можно заранее вычислить все значения $b^\delta \mod n$ для $\delta \in D$ и сохранить полученные числа в массиве. Вторая стадия алгоритма выполняется следующим образом:

1. Вычислим сначала $c_0 = b^{q_0} \mod n$, и найдем d =Н.О.Д.$(n, _0 - 1)$.

2. Если $d = 1$, то вычислим следующее $c_1 = b^{q_1} \mod n$ и d =Н.О.Д.$(n, _1 - 1)$ и т.д.

3. Каждое последующее значение c_{i+1} вычисляется по формуле

$$b^{q_{i+1}} \bmod n = b^{q_i+\delta_i} \bmod n = b^{q_i} \cdot b^{\delta_i} \bmod n = c_i \cdot b^{\delta_i} \bmod n. \tag{5.4.6}$$

Поскольку все значения $b^{\delta_i} \bmod n$ заранее вычислены, то для вычисления очередного значения c_{i+1} достаточно одной операции умножения и вычисления остатка по модулю n. Поэтому вторая стадия алгоритма Полларда выполняется очень быстро.

Оценка эффективности $(p-1)$–метода Полларда

Сделаем расчет времени работы алгоритма при условии, что параметры B_1 и B_2 выбраны. Время выполнения первой стадии зависит от числа простых чисел и их степеней на интервале $[2; B]$. Число простых чисел оценивается величиной $\pi(B_1)$, приближенно равной по теореме Чебышева числу $B_1/\ln B_1$. Для каждой степени p^r, меньшей B, производится r возведений в степень по модулю по алгоритму, описанному на с. 63, и требует $\log_2 p \le \log_2 B_1$ операций возведения в квадрат и умножений по модулю числа n. Поэтому общее число операций можно оценить величиной $O(B_1 \log B_1 \log^2 N)$. Метод очень быстро находит простые факторы малой и средней величины (до 20-25 десятичных цифр). Текущим рекордом для $(p-1)$–метода является простой делитель числа $960^{119} - 1$, состоящий из 66 десятичных цифр, установленный T. Нохара (T. Nohara) в 2006 г.

Использование второй стадии позволяет увеличить эффективность метода. По оценке Монтгомери, вторая стадия алгоритма требует

$$O(\log^2 B_2) \; + \; O(\log q_{\pi(B_1)}) \; + \; 2(\pi(B_2) - \pi(B_1))$$

умножений по модулю n и вычислений Н.О.Д. с n. Отбрасывая слагаемые меньшего порядка, получим оценку $O(\pi(B_2))$.

Условие сходимости $(p-1)$–метода Полларда

Пусть p–наименьший из делителей n и q^t–наибольшая степень простого числа, входящего в разложение $p-1$. Иначе говоря, q^t максимально среди всех степеней $q_i^{t_i} \mid p-1$. Отметим, что оценка сложности $(p-1)$–

алгоритма определяется не размером факторизуемого числа n, а размером сомножителя q^t чисел $p-1$ для $p \mid n$.

Если $q^t \leq B_1$, тогда вычисление закончится на первом этапе алгоритма. Иначе, для успеха алгоритма необходимо, чтобы выполнилось $q^t \leq B_2$, а все степени простых делителей $(p-1)$ вида q^r кроме последнего были меньше B_1. Кроме того необходимо, чтобы среди делителей $p-1$ не нашлось множителей вида r^k при $k \geq 2$, находящегося между B_1 и B_2. Для таких r^k имеем два неравенства:

$$r^{k-1} \leq B, \quad B_1 < r^k < B_2,$$

откуда

$$B_1^{1/k} < p < min\{B^{1/(k-1)},\ B_2^{1/k}\}. \tag{5.4.7}$$

При $B_2 = cB_1$ и $k = 2$ уравнения (5.4.7) приобретут вид:

$$\sqrt{B_1} < r < min\{B_1,\ \sqrt{cB_1}\}.$$

Поскольку значение границы B_2 обычно выбирается так, чтобы выполнялось $B_2 \leq B_1^2$, последнее уравнение эквивалентно

$$\sqrt{B_1} < r \ < \ c_1\sqrt{B_1}, \text{ где } c_1 = \sqrt{c}. \tag{5.4.8}$$

Общая доля чисел, удовлетворяющих (5.4.8), невелика и значительно меньше числа простых чисел из интервала $(B_1,\ B_2)$, поэтому ими можно пренебречь. Однако, если не пренебрегать этими числами и добавить в алгоритм дополнительный цикл по элементам r, удовлетворяющим условию (5.4.8), общее время алгоритма увеличится незначительно. Поскольку размер наибольшей степени q^t сильно зависит от степени гладкости числа $p-1$, поэтому эффективность $(p-1)$–метода Полларда сильно зависит от исходного числа n, изменяясь в широких пределах при различных n одинаковой длины. Поэтому одной из рекомендаций метода RSA является выбор n так, чтобы $p-1$ имел хотя-бы один большой делитель, превышающий размер границы B_2, до которой возможно выполнение реальных вычислений по $(p-1)$–методу Полларда.

Скажем несколько слов о выборе исходного значения параметра a. Если $p-1$ имеет большое число различных делителей, то найдется много различных чисел $a < n$, для которых $a^k \equiv 1 \,(\mathrm{mod}\, p)$ выполнится для $k < p-1$. Найдутся даже такие $a < n$, что уже $a^2 \equiv 1 \,(\mathrm{mod}\ p)$. Для таких a скорость схождения метода будет значительно выше. Поэтому, можно ускорить сходимости $(p-1)$–метода Полларда, запуская алгоритм с

несколькими значениями a. В статье «Pollard ρ on the Play Station 3», размещенной на сайте http://www.hyperelliptic.org/tanja/SHARCS/slides09/03-bos.pdf, приводятся примеры программирования алгоритмов факторизации, включая ρ и $(p-1)$ методы Полларда с возможностью распараллеливания на игровой консоли Sony Play Station 3, имеющей 8 сопроцессоров.

Пример

Пусть $p = 29$–делитель n, тогда, $p-1 = 28 = 2^2 \cdot 7$. Для каждого $a \leq 28$ найдем наименьшее k такое, что $a^k \equiv 1 \pmod{p}$. Приведем фрагмент полученной таблицы:

a	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
k	28	28	14	14	14	7	28	14	28	28	4	14	28	28	7	4

Среди 28 значений $a < 29$ окажется 12 значение с показателем $k = 28$, по 6 значений с показателем $k = 14$ и $k = 7$, 2 значения с $k = 4$, по одному с $k = 2$ и $k = 1$. Математическое ожидание наименьшего показателя k равно:

$$M[k] = (12 \cdot 28 + 6 \cdot 14 + 6 \cdot 7 + 2 \cdot 4 + 2 \cdot 1)/28 \approx 16,85$$

Таким образом, выбирая удачное $a < n$, можно получить значительный выигрыш по сравнению с произвольным.

Дальнейшие улучшения алгоритма

Для ускорения всех вычислений Полларда предложил использовать быстрое преобразование Фурье (a Fast Fourier Transform) для всех основных операций.

Дальнейшие улучшения алгоритма были предложены П.Монтгомери [31]. Он заметил, что на второй стадии алгоритма большую часть времени отнимает вычисление для каждого простого числа q_i из интервала $[B_1;\ B_2]$ Н.О.Д.$(n, {}_i - 1)$, где $c_i = b^{q_i}$. Монтгомери предложил объединять несколько соседних элементов c_i в блоки G_j и вычислять сначала произведение $h = \prod(c_i - 1) \mod n$ для всех $c_i \in G_j$, а потом Н.О.Д.(n, h). Если Н.О.Д.$(n, {}_i - 1) > 1$, то таковым будет и Н.О.Д.(n, h). Эти и другие улучшения, найденные Монтгомери, позволяют в несколько раз сократить общее время работы алгоритма.

На сайте www.loria.fr/∼zimmerma/records/Pminus1.html можно найти таблицу рекордов разложения натуральных чисел, установленных с помощью $(p-1)$–метода Полларда.

Упражнение.

Сформируйте множество E_n всех составных чисел n одинаковой длины, являющихся произведением двух простых чисел. Найдите для каждого числа n математическое ожидание $M[k]$ наименьшего показателя k для множества элементов $a < p$, где p– наименьший делитель n. Распределите все числа из множества E_n в классы в зависимости от значения $M[k]$. Сделайте вывод о доле составных чисел, которые могут быть быстро разложены в произведение простых сомножителей с помощью $(p-1)$–метода Полларда.

В разделе "Метод факторизации Ленстры" мы увидим, как идея $(p-1)$-метода Полларда была использована Х.Ленстрой для построения нового более быстрого метода факторизации с использованием эллиптических кривых.

5.5 $(p+1)$–метод Вильямса

Определение. Последовательностью Люка (Lucas) назовем рекуррентную последовательность u_n, определяемую соотношениями:

$$u_0 = 0, \quad u_1 = u, \quad u_{n+1} = P \cdot u_n - Q \cdot u_{n-1}, \tag{5.5.9}$$

где P, Q – фиксированные целые числа.

$(p+1)$–метод Вильямса (Williams) похож на $(p-1)$–метод Полларда и основан на предположении гладкости числа $p+1$. Пусть p–простой делитель факторизуемого числа n, и выполнено разложение $p+1$

$$p+1 = \prod_{i=1}^{k} q_i^{a_i}.$$

Обозначим через $B = \max\{q_i^{a_i} | 1 \le i \le k\}$. По-прежнему будем называть натуральное число r B–степенно-гладким, если наибольшая степень сомножителя $p_i^{a_i}$ в разложении r на простые множители, не превышает B. Таким образом, определенное выше число B является наименьшим числом, для которого $p+1$ является B–степенно-гладким. Отметим, что поскольку p не известно, то и B так же не известно.

Алгоритм Вильямса заключается в следующем:

1. Выбираем некоторое число B, являющее верхней границей для рассматриваемых простых чисел и их степеней.

2. Строим последовательность простых чисел $2 < 3 < 5 < ... < p_m$, меньших B и последовательность степеней a_i такую, что $p_i^{a_i} < B$.

3. Полагаем число $R = \prod_{i=1}^{m} q_i^{a_i}$. Если p является B–степенно-гладким, то R делится на p.

4. Выбираем случайным образом числа P и Q и строим последовательность чисел Люка, пока не вычислим u_R.

5. Далее вычислим Н.О.Д.$(n, u_R) = d$. Если $1 < d < n$, то задача решена.

Доказано, что если Q взаимно просто с p и

$$\left(\frac{P^2 - 4Q}{p}\right) = -1,$$

то свойства последовательности Люка обеспечивают нахождение нетривиального делителя числа n.

5.6 ρ-метод Полларда

Этот метод был разработан Джоном Поллардом в 1975 г. Пусть n – число, которое следует разложить. ρ-метод Полларда работает следующим образом:

1. Выбираем небольшое число x_0 и строим последовательность чисел $\{x_n\},\ n = 0,\ ,1,\ 2,\ ...,$ определяя каждое следующее x_{n+1} по формуле $x_{n+1} = (x_n^2 - 1)\,(\mathrm{mod}\, n)$.

2. Одновременно на каждом шаге i вычисляем Н.О.Д d числа n и всевозможных разностей $|x_i - x_j|$, где $j < i$.

3. Когда будет найдем d =Н.О.Д.$(n,\ |x_i - x_j|)$, отличный от 1, вычисление заканчивается. Найденное d является делителем n. Если n/d не является простым числом, то процедуру можно продолжить, взяв вместо n число n/d.

Вместо функции $F(x) = (x^2 - 1) \bmod n$ в вычислении x_{n+1} можно взять другой многочлен, например, $x^2 + 1$ или произвольный многочлен 2-й степени $F(x) = ax^2 + bx + c$.

Недостатком данного варианта метода является необходимость хранить большое число предыдущих значений x_j. Заметим, что если

$(x_j - x_i) \equiv 0\,(\,\mathrm{mod}\, p),\ \text{то}\ (f(x_j) - f(x_i)) \equiv 0\,(\,\mathrm{mod}\, p),$

поэтому, если пара (x_i, x_j) дает нам решение, то решение даст любая пара (x_{i+k}, x_{j+k}).

Поэтому, нет необходимости проверять все пары (x_i, x_j), а можно ограничиться парами виды (x_i, x_j), где $j = 2^k$, и k пробегает набор последовательных значений 1, 2, 3, ..., а i принимает значения из интервала $[2^k + 1; 2^{k+1}]$. Например, при $k = 3$ $j = 2^3 = 8$, а $i \in [9; 16]$.

```
int ρ-Pollard (int n)
{ int x = random (1, n-2);
int y = 1; int i = 0; int stage = 2;
while(Н.О.Д.(n, abs(x − y)) = 1)
{
   if (i = = stage ){
      y = x;
       stage = stage*2; }
   x=x ∗ x + 1(mod n);
   i=i + 1;
}
return Н.О.Д.(n, abs(x − y)); }
```

В этом варианте вычисление требует хранить в памяти всего три переменные n, x и y, что выгодно отличает этот метод от других методов факторизации.

Еще одна вариация ρ-метода Полларда была разработана Флойдом (Floyd). Согласно Флойду, значение y обновляется на каждом шаге по формуле $y = F^2(y) = F(F(y))$, поэтому на шаге i будут получены значения $x_i = F^i(x_0)$, $y_i = x_{2i} = F^{2i}(x_0)$, и Н.О.Д. на этом шаге вычисляется между n и $y - x$.

Обоснование ρ-метода Полларда

Приведем обоснование этого метода и оценим его трудоемкость. Оценка основывается на известном «парадоксе дня рождения».

Теорема 5.6.1 *(Парадокс дня рождения) Пусть $\lambda > 0$. Для случайной выборки из $l+1$ элементов, каждый из которых меньше q, где $l = \sqrt{2\lambda q}$, вероятность того, что два элемента окажутся равными*

$$p > 1 - e^{-\lambda}.$$

Отметим, что вероятность $p = 0,5$ в парадоксе дня рождения достигается при $\lambda \approx 0,69$.

Пусть последовательность $\{u_n\}$ состоит из разностей $|x_i - x_j|$, проверяемых в ходе работы алгоритма. Определим новую последовательность $\{z_n\}$, где $z_n = u_n \bmod q$, q – меньший из делителей n. Все члены последовательности $\{z_n\}$ меньше $\sqrt{n}$. Если рассматривать $\{z_n\}$ как случайную последовательность чисел, меньших q, то, согласно парадоксу близнецов, вероятность того, что среди первых $l+1$ ее членов попадутся два одинаковых, превысит $1/2$ при $\lambda \approx 0,69$, тогда l должно быть не меньше $\sqrt{2\lambda q} \approx \sqrt{1.4q} \approx 1,18\sqrt{q}$.

Если $z_i = z_j$, тогда $x_i - x_j \equiv 0 \bmod q \to x_i - x_j = kq$ для некоторого $k \in \mathbf{Z}$. Если $x_i \neq x_j$, что выполняется с большой вероятностью, то искомый делитель q числа n будет найден как Н.О.Д.$(n, x_i - x_j)$. Поскольку $\sqrt{q} \leq n^{1/4}$, то с вероятностью, превышающей 0,5, делитель n может быть найден за $1,18 \cdot n^{1/4}$ итераций.

Таким образом, ρ-метод Полларда является вероятностным методом, позволяющим найти нетривиальный делитель q числа n за $O(q^{1/2}) \leq O(n^{1/4})$ итераций. Сложность вычисления нетривиального делителя в этом методе зависит только от размера этого делителя, а не от размера числа n. Поэтому, ρ–метод Полларда применим в тех случаях, когда другие методы факторизации, зависящие от размера n, становятся не эффективными.

Отметим, что в некоторых случаях, последовательность $\{y_n\}$ может зацикливаться (т.е. на некотором шаге t появляется $x_t = x_0$, после чего последовательность повторяется), тогда надо поменять исходный элемент x_0 или полином $F(x)$ на какой–нибудь другой.

Упражнения.

1. Подберите несколько составных чисел n одинаковой длины, и выполните пробное разложение этих чисел методом Полларда. Вычислите среднее время (количество итераций) до нахождения нетривиального делителя n. Как сильно отличается время вычисления для различных n?

2. Выполните упражнение 1 с алгоритмом Флойда. Сравните среднее время вычисления делителя в первом и втором случаях.

5.7 ρ-метод Полларда для вычисления дискретного логарифма

Проблема дискретного логарифма (Discrete Logarithm Problem DLP) состоит в вычислении в конечном поле F_q с образующей g для произвольного элемента t наименьшего числа k такого, что $g^k = t$. Хотя эта проблема не связана непосредственно с проблемой факторизации целых чисел, она играет важную роль в криптографии. При длине ключа L проблема DLP имеет такую же сложность решения, как и проблема факторизации числа длины L, поэтому на проблеме вычисления DLP построено много криптографических протоколов, в том числе, известные протоколы Диффи-Хелмана вычисления общего секретного ключа и Эль-Гамаля электронной цифровой подписи.

Существует большое число различных методов для решения этой задачи. В главе 5 книги Л.Вашингтона [39] дано описание основных алгоритмов для ДЛЭК. В этом разделе мы рассмотрим ρ-метод Полларда для DLP, который играет здесь ту же роль, что и ρ-метод Полларда для проблемы факторизации. Группу по умножению поля F_p, p–простое число, обозначим через $F_p^* = \{1,\, 2 \ldots p-1\}$. Напомним, что элемент $g \in F_p^*$ называется генератором поля, если любой элемент $t \in F_p^*$ равен некоторой степени элемента g: $t = g^k$. Пусть g – (какой-нибудь) генератор этой группы, и пусть t - произвольный элемент F_p^*.

Для нахождения неизвестного показателя k такого, что $g^k = t$, будем строить последовательность пар $(a_i,\, b_i)$ чисел по модулю $p-1$ и последовательность x_i чисел по модулю p такую что $x_i = t^{a_i} g^{b_i}$. Определим начальные значения $a_0 = b_0 = 0$, $x_0 = 1$. Вычисление последующих членов последовательностей будем выполнять по формулам:

$$(a_{i+1}, b_{i+1}) = \begin{cases} (a_i + 1,\, b_i) \bmod (p-1), & \text{если } 0 < x_i < p/3, \\ (2a_i,\, 2b_i) \bmod (p-1), & \text{если } p/3 < x_i < 2p/3, \\ (a_i,\, b_i + 1) \bmod (p-1) & \text{если } 2p/3 < x_i < p, \end{cases} \tag{5.7.10}$$

и, соответственно,

$$x_{i+1} = \begin{cases} t x_i \bmod p, & \text{если } 0 < x_i < p/3, \\ x_i^2 \bmod p, & \text{если } p/3 < x_i < 2p/3, \\ g x_i \bmod p & \text{если } 2p/3 < x_i < p, \end{cases} \tag{5.7.11}$$

Эта последовательность вычисляется до тех пор, пока не появятся номера

i, j такие, что $x_i = x_j$. Тогда, $t^{a_i} g^{b_i} = t^{a_j} g^{b_j}$, откуда,

$$(a_j - a_i)k \equiv b_i - b_j \pmod{(p-1)} \tag{5.7.12}$$

Если Н.О.Д.$(a_j - a_i,\ p-1) = 1$, тогда множитель k в уравнении (5.7.12) может быть найден с использованием обобщенного алгоритма Евклида, решив в целых числах уравнение

$$x(a_j - a_i) + y(p-1) = b_i - b_j \tag{5.7.13}$$

относительно x, y и определяя $k = x \mod (p-1)$.

Если же Н.О.Д.$(a_j - a_i,\ p-1) = d > 1$, тогда, уравнение (5.7.13) попрежнему, разрешимо, но дает решение нашего уравнения с точностью до слагаемого кратного $(p-1)/d$, т.е. решение имеет вид

$$x = x_0 + m(p-1)/d \tag{5.7.14}$$

где $m \in [0, d-1]$–целое число. Если множитель d - мал, то решение будет найдено подстановкой чисел (5.7.14) в уравнение $g^X \equiv t \mod t$.

Так же, как в ρ–методе факторизации, в этом алгоритме можно использовать модификацию Флойда, вычисляя на i-м шаге одновременно тройку (a_i, b_i, x_i) и тройку (a_{2i}, b_{2i}, x_{2i}), пока не дойдем до шага i, на котором $x_i = x_{2i}$. В этом варианте опять не надо хранить в памяти на шаге i все тройки (a_j, b_j, x_j) для $j \le i$, а достаточно сохранять две тройки (a_i, b_i, x_i) и (a_{2i}, b_{2i}, x_{2i}).

Пример. Рассмотрим поле F_p при $p = 43$. Элемент $g = 2$ не является генератором по критерию Поклингтона, т.к.$2^{14} \bmod 43 = 1$. Возьмем в качестве генератора элемент $g = 3$ и решим уравнение

$$3^X \bmod 43 = 15. \tag{5.7.15}$$

Итак, $p = 43$, $g = 3$, $t = 15$. Определим $(_0, b_0, x_0) = (0, 0, 1)$, и будем строить две последовательности (a_i, b_i, x_i) и (a_{2i}, b_{2i}, x_{2i}) по формулам (5.7.10) и (5.7.11):

i	a_i	b_i	x_i	a_{2i}	b_{2i}	x_{2i}
1	2	0	10	6	0	11
2	3	0	21	7	1	22
3	6	0	11	15	2	36
4	7	0	36	30	6	11
5	7	1	22	31	7	22

На 5-м шаге значения x_i и x_{2i} совпали. Составим уравнение (5.7.13):

$$x(31-7)+y(43-1) \equiv 1-7 \,(\mod 42), \quad \text{или,} \quad 24x+42y \equiv 36 \,(\mod 42).$$

Вычислим Н.О.Д.$(a_j - a_i,\, p-1)$ =Н.О.Д.$(24,\, 42) = 6 \neq 1$.

Поделим все коэффициенты уравнения на 6:

$$4x + 7y \equiv 6 \,(\mod 42).$$

С помощью расширенного алгоритма Евклида решим уравнение $4x + 7y = 1$, подставляя вместо A и B коэффициенты этого уравнения (поменяв их местами, чтобы A было больше B):

A	B	A mod B	A div B	y	x
7	4	3	1	-1	2
4	3	1	1	1	-1
3	1	0	3	0	1

Таким образом, $7 \cdot (-1) + 4 \cdot 2 = 1$, или, $7 \cdot (-6) + 4 \cdot 12 = 6$. Положим, $x_0 = x = 12$. Поскольку, $d > 1$, то корень X уравнения (5.7.15) определяется с точностью до слагаемое $(p-1)/d = 7$, т.е. имеет вид $X = x_0 + 7k$, где $k \in \mathbf{Z}$. Будем подставлять в (5.7.15) последовательно числа 12, 19, 25, ..., пока не получим тождество $3^{25} \mod 43 = 15$. Задача решена.

Глава 6

Криптографические методы, основанные на задаче дискретного логарифмирования в конечном поле

Напомним, что в конечном поле F_q, $q = p^k$, для произвольных элементов $a, b \in F_q$ дискретным логарифмом числа b по основанию a называется натуральное число k такое, что

$$a^k = b \text{ в поле } F_q$$

Если степень расширения поля $k = 1$ т.е. $q = p$ является простым числом, то последнее условие можно переписать в виде

$$a^k \bmod p = b$$

Задача вычисления дискретного логарифма при заданных значениях p, a, b называется задачей дискретного логарифмирования в конечном поле ДЛП.

Существует тривиальный переборный алгоритм вычисления степени k путем сравнения элементов $a^x \bmod p$ и b для $x = 0, 1, 2, \ldots$ до их совпадения. Этот алгоритм работает очень медленно и имеет сложность экспоненциальную сложность относительно длины L размерности q.

В первой главе был описан ρ-алгоритм Полларда, основанный на парадоксе дня рождения, имеющий сложность $O(2^{L/2})$. Самым быстрым на сегодняшний день является алгоритм *решета числового поля*, имеющий субэкспоненциальную сложность.

Принято считать, что длина размерности поля, при которой дискретный логарифм не вычислим, равна 1024 бита или более, что соответствует длине числа $n = p \cdot q$ в методе RSA. Поэтому методы, использующие трудноразрешимость задачи ДЛП, имеют ключи той же длины, что и метод RSA, основанный на трудноразрешимость задачи факторизации.

6.1 Протокол Диффи-Хеллмана

Этот протокол предназначен для формирования общего секретного ключа для двух удаленных пользователей, общающихся через открытые сети. Обозначим этих пользователей буквами A и B.

Алгоритм протокола Диффи-Хеллмана.

I. Первая часть протокола состоит в генерации ключей – простого числа p длины 1024 бита или более, и нахождения генератора группы по умножению поля F_p. Напомним, что элемент a, $1 < a < p$, называется генератором (порождающим элементом), если любой другой ненулевой элемент $b \in F_p$, $b \neq 0$, является степенью элемента a.

Пример. Пусть $p = 13$. Вычислим все степени элементов 2 и 3 в поле F_{13}:

k	1	2	3	4	5	6	7	8	9	10	11	12
$2^k \bmod 13$	2	4	8	3	6	12	11	9	5	10	7	1
$3^k \bmod 13$	3	9	1	3	9	1	3	9	1	3	9	1

Таким образом, 2 является генератором поля, а 2 – нет. Для поиска простого числа p можно воспользоваться тестом Миллера-Рабина. Выбирается произвольное нечетное простое число t заданной длины, проверяется условие $t \bmod q \neq 0$ для всех небольших простых чисел, затем выполняется несколько раундов проверок алгоритма Миллера-Рабина. Если число не проходит этот тест, то рассмотрим следующее число $t + 2$ и т.д.

Для поиска порождающего элемента можно проверять все элементы подряд, начиная от 2, непосредственным возведением в степень, либо используя тест Поклингтона.

После того, как число p и генератор g поля F_p найдены, эти параметры распространяются среди участников протокола. Параметры p и a не являются секретными, и их передача выполняется по открытому каналу. Эти параметры могут быть использованы многократно.

II. Вторая часть состоит в выработке общего секретного ключа и состоит из следующих шагов:

1. Участник A выбирает случайное число $1 < a < p$, а участник B – $1 < b < p$. Потом участники вычисляют $x = g^a \mod p$ и $y = g^b \mod p$ и пересылает их открыто другому участнику.

2. После обмена числами участники вычисляют общий ключ k по

формулам

$$A: \ y^a \mod p = g^{ba} \mod p = k$$
$$B: \ x^b \mod p = g^{ab} \mod p = k$$

Противник, перехватывая пересылаемые данные, получит в свое распоряжение параметры $< p, g, x = g^a, y = g^b >$, которых будет недостаточно для вычисления k. Для вычисления k необходимо найти параметры a или b, решая проблему ДЛП.

Этот протокол уязвим к атаке "человек-посередине", которая возможна, если противник имеет возможность перехватывать сообщения от A к B и обратно и заменять их своими данными. Тогда секретный ключ будет сформирован противником, который поучит доступ к переписке.

Для исправления ситуации в протокол необходимо добавить процедуру аутентификации участников с использованием *электронной цифровой подписи.*

Описание понятия электронной цифровой подписи в следующем параграфе.

6.2 Электронная цифровая подпись и ее свойства

Одной из наиболее важных задач информационной безопасности является сохранение целостности (неизменности) электронных документов, передаваемых через каналы связи. Развитие современных средств безбумажного документооборота, средств электронных платежей немыслимо без развития средств доказательства подлинности и целостности документа. Таким средством является электронная цифровая подпись (ЭЦП), которая сохранила основные свойства обычной подписи. Такими свойства подписи являются:

1. Аутентичность – ЭЦП является доказательством того факта, что подпись принадлежит подписывающему;
2. Подпись неподделываема; то есть служит доказательством того, что только тот человек, чей автограф стоит на документе, мог подписать данный документ, и никто иной.
3. Подпись непереносима, т.к. является частью документа и поэтому перенести ее на другой документ невозможно.
4. Документ с подписью является неизменяемым.

5. Подпись неоспорима – любое лицо, владеющее образцом подписи может удостоверится, что документ подписан владельцем подписи.

Существует несколько методов построения ЭЦП, а именно: шифрование электронного документа (ЭД) на основе симметричных алгоритмов. Данная схема предусматривает наличие в системе третьего лица-арбитра, пользующегося доверием обеих сторон. Авторизацией документа в данной схеме является сам факт шифрования ЭД секретным ключом и передача его арбитру. Использование ассиметричных алгоритмов шифрования. Фактом подписания документа является шифрование его на секретном ключе отправителя.

Развитием предыдущей идеи стала наиболее распространенная схема ЭЦП - шифрование окончательного результата обработки ЭД хеш-функцией при помощи асимметричного алгоритма. Кроме перечисленных, существуют и другие методы построения схем ЭЦП - групповая подпись, неоспариваемая подпись, доверенная подпись и др. Появление этих разновидностей обусловлено разнообразием задач, решаемых с помощью электронных технологий передачи и обработки электронных документов.

Электронная цифровая подпись используется физическими и юридическими лицами в качестве аналога собственноручной подписи для придания электронному документу юридической силы, равной юридической силе документа на бумажном носителе, подписанного собственноручной подписью правомочного лица и скрепленного печатью. ЭЦП - это программно-криптографическое средство, которое обеспечивает решение следующих задач:

1. проверку целостности документов;
2. идентификация лица, отправившего документ;

Следует отметить, что конфиденциальность документа, т.е. скрытие его содержимого от третьих лиц, электронная цифровая подпись не обеспечивает.

Правовые основы электронной цифровой подписи регламентируются несколькими законами Российской Федерации. В частности, пункт 3 статьи 5 Федерального закона "Об информации, информатизации и защите информации"гласит: "Юридическая сила документа, хранимого, обрабатываемого и передаваемого с помощью автоматизированных информационных и телекоммуникационных систем, может подтверждаться электронной цифровой подписью. 10 января 2002 года Президентом

РФ был подписан очень важный закон "Об электронной цифровой подписи"номер 1-ФЗ (принят Государственной Думой 13 декабря 2001 года), развивающий и конкретизирующий основные положения закона "Об информации, информатизации и защите информации". Его роль поясняется в статье 1. 1. Целью настоящего Федерального закона является обеспечение правовых условий использования электронной цифровой подписи в электронных документах, при соблюдении которых электронная цифровая подпись в электронном документе признается равнозначной собственноручной подписи в документе на бумажном носителе. 2. Действие настоящего Федерального закона распространяется на отношения, возникающие при совершении гражданско-правовых сделок и в других предусмотренных законодательством Российской Федерации случаях. Действие настоящего Федерального закона не распространяется на отношения, возникающие при использовании иных аналогов собственноручной подписи.

Закон вводит *следующие* основные понятия:

Электронный документ - документ, в котором информация представлена в электронно-цифровой форме.

Электронная цифровая подпись - реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной цифровой подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе.

Владелец сертификата ключа подписи - физическое лицо, на имя которого удостоверяющим центром выдан сертификат ключа подписи и которое владеет соответствующим закрытым ключом электронной цифровой подписи, позволяющим с помощью средств электронной цифровой подписи создавать свою электронную цифровую подпись в электронных документах (подписывать электронные документы).

Средства электронной цифровой подписи - аппаратные и (или) программные средства, обеспечивающие реализацию хотя бы одной из следующих функций: создание электронной цифровой подписи в электронном документе с использованием закрытого ключа электронной цифровой подписи, подтверждение с использованием открытого ключа электронной цифровой подписи подлинности электронной цифровой подписи в электронном документе, создание закрытых и открытых ключей элек-

тронных цифровых подписей.

Сертификат средств электронной цифровой подписи - документ на бумажном носителе, выданный в соответствии с правилами системы сертификации для подтверждения соответствия средств электронной цифровой подписи установленным требованиям.

Закрытый ключ электронной цифровой подписи - уникальная последовательность символов, известная владельцу сертификата ключа подписи и предназначенная для создания в электронных документах электронной цифровой подписи с использованием средств электронной цифровой подписи.

Открытый ключ электронной цифровой подписи - уникальная последовательность символов, соответствующая закрытому ключу электронной цифровой подписи, доступная любому пользователю информационной системы и предназначенная для подтверждения с использованием средств электронной цифровой подписи подлинности электронной цифровой подписи в электронном документе.

Сертификат ключа подписи - документ на бумажном носителе или электронный документ с электронной цифровой подписью уполномоченного лица удостоверяющего центра, которые включают в себя открытый ключ электронной цифровой подписи и выдаются удостоверяющим центром участнику информационной системы для подтверждения подлинности электронной цифровой подписи и идентификации владельца сертификата ключа подписи.

Подтверждение подлинности электронной цифровой подписи в электронном документе - положительный результат проверки соответствующим сертифицированным средством электронной цифровой подписи с использованием сертификата ключа подписи принадлежности электронной цифровой подписи в электронном документе владельцу сертификата ключа подписи и отсутствия искажений в подписанном данной электронной цифровой подписью электронном документе.

Пользователь сертификата ключа подписи - физическое лицо, использующее полученные в удостоверяющем центре сведения о сертификате ключа подписи для проверки принадлежности электронной цифровой подписи владельцу сертификата ключа подписи.

Информационная система общего пользования - информационная система, которая открыта для использования всеми физическими и юридическими лицами и в услугах которой этим лицам не может быть отказано.

Корпоративная информационная система - информационная система, участниками которой может быть ограниченный круг лиц, определенный ее владельцем или соглашением участников этой информационной системы.

Юридическая сила электронной цифровой подписи признается при наличии в автоматизированной информационной системе программно-технических средств, обеспечивающих идентификацию подписи, и соблюдении установленного режима их использования". Такая формулировка предполагает, что электронный документ может быть заверен ЭЦП и использован в тех случаях, когда явно не предусмотрены другие требования к форме документа, т.е. введение данной нормы, по существу, не расширило возможности использования ЭДО в гражданском обороте. Развитие основных типов криптографических протоколов (ключевой обмен, электронная цифровая подпись (ЭЦП), аутентификация и др.) было бы невозможно без создания открытых ключей и построенных на их основе асимметричных протоколов шифрования. Эти методы рассматриваются в следующем разделе.

6.3 Односторонние функции. Хеш-функции

Односторонняя (однонаправленная) функция (one way function) - это отображение $X \to Y$, где X и Y - произвольные множества, удовлетворяющее следующим условиям:

1. Для каждого x из области определения функции f легко вычислить $f(x)$. Понятие "легко"обычно означает, что существует алгоритм, вычисляющий функцию $f(x)$ за полиномиальное время от длины аргумента x.

2. Задача нахождения прообраза x для произвольного y, принадлежащего области значений функции f, является вычислительно сложной задачей. Последнее означает, что не существует алгоритма, вычисляющего существенно быстрее, чем алгоритм полного перебора.

Пример 1. Задача вычисления простых множителей натурального числа n (факторизация N) является односторонней функций – самый быстрый известный алгоритм факторизации – метод решета числового поля имеет субэкспоненциальную верхнюю оценку времени своей работы.

На вычислительной сложности решения этой задачи построен один из самых известных асимметричных методов криптографии - метод RSA.

Пример 2. Примером односторонней функции может служить вычисление функции $f(x) = a^x \mod n$, где a и n - некоторые фиксированные натуральные числа. Задача вычисления обратного значения x по известному $f(x)$ называется задачей дискретного логарифмирования. На вычислительной сложности задачи дискретного логарифмирования основан один из распространенных методов двухключевой криптографии - метод Эль-Гамаля.

Пример 3. Функция $f(k) = [k]P$ вычисления кратного точки p эллиптической кривой: даны конечное поле GF_q, эллиптическая кривая E над полем GF_q, точка P на кривой E. По известным координатам точки P и аргументу k функция f вычисляет координаты т. $[k]P$. Эта задача полиномиально вычислима.

Обратная задача нахождения координат т.P по известным k и координатам т.$[k]P$ является вычислительно сложной задачей, имеющей субэкспоненциальный алгоритм вычисления.

Пример 4. Примером использования односторонней функции может служить следующая схема аутентификации. Абоненты A и B договариваются при встрече или вырабатывают с помощью протокола Диффи-Хеллмана общий ключ x. Теперь когда абоненты выходят на связь, то один из них, скажем A, посылает другому послание M, некоторое число $k > 2$ и число y, равное результату применения к аргументу x k-кратной итерации односторонней функции $f(x)$:

$$y = f^{(k)}(x) = f(f(\ \ldots(x)\ldots)$$

Абонент B, получив число k, также вычисляет значение $y = f^{(k)}(x)$ и сравнивает его с полученным. Если результат совпал, то сообщение получено именно от абонента A. Абонент B, возвращая ответное послание $_2$, прикладывает к нему значение $y_{k-1} = f^{(k-1)}(x)$. Взломщик не может подделать сообщение y_{k-1}, т.к. даже зная $f^{(k)}(x)$, он не может вычислить $y = f^{(k-1)}(x)$. При следующем обмене пересылается число $y_{k-2} = f^{(k-2)}(x)$, и т.д., что обеспечивает взаимную аутентификацию при каждом новом сеансе связи.

Хеш-функции

Хеш-функции играют в информационной защите важную роль, создавая для электронного документа его "моментальный снимок"и тем самым защищая документ от дальнейшей модификации или подмены. В широком смысле функцией хеширования называется функция H, удовлетворяющая следующим основным свойствам:

1. Хеш-функция может применяться к блоку данных любой длины.
2. Хеш-функция создает выход фиксированной длины (равно, например, 128 бит для классической функции хеширования MD5, и 160 бит для американского стандарта хеш-функций SHA1).
3. Значение $= h(M)$ вычисляется относительно быстро (за полиномиальное время от длины сообщения M).
4. Для любого данного значения хеш-кода H вычислительно невозможно найти M такое, что $h(M) = H$.
5. Для любого данного y вычислительно невозможно найти x такое, что $y = h(x)$.
6. Вычислительно невозможно найти произвольную пару $(, y)$ такую, что $H(y) = H(x)$.

Термин вычислительно невозможно означает здесь, что в настоящее время решение этой задачи либо требует слишком большого интервала времени (например, более сотни лет), либо использования слишком больших вычислительных ресурсов, чтобы решение задачи имело смысл. Первые три свойства требуют, чтобы хеш-функция создавала хеш-код для любого сообщения. Четвертое свойство определяет требование одностороннести хеш-функции: легко создать хеш-код по данному сообщению, но невозможно восстановить сообщение по данному хеш-коду. Это свойство важно, если аутентификация с использованием хеш-функции включает секретное значение. Само секретное значение может не посылаться, тем не менее, если хеш-функция не является односторонней, противник может легко раскрыть секретное значение следующим образом. При перехвате передачи атакующий получает сообщение M и хеш-код $H = h(S_{AB}||M)$. Если атакующий может инвертировать хеш-функцию, то, следовательно, он может получить $S_{AB}||M = H^{-1}(C)$. Так как атакующий теперь знает и M, и $S_{AB}||M$, получить S_{AB} совсем просто, здесь

|| обозначает операцию конкатенации (соединения) двух текстов. Пятое свойство гарантирует, что невозможно найти другое сообщение, чье значение хеш-функции совпадало бы со значением хеш-функции данного сообщения. Это предотвращает подделку аутентификатора при использовании зашифрованного хеш-кода. В данном случае противник может читать сообщение и, следовательно, создать его хеш-код. Но так как противник не владеет секретным ключом, он не имеет возможности изменить сообщение так, чтобы получатель этого не обнаружил . Если данное свойство не выполняется, атакующий имеет возможность выполнить следующую последовательность действий: перехватить сообщение и его зашифрованный хеш-код, вычислить хеш-код сообщения, создать альтернативное сообщение с тем же самым хеш-кодом, заменить исходное сообщение на поддельное. Поскольку хеш-коды этих сообщений совпадают, получатель не обнаружит подмены. Хеш-функция, которая удовлетворяет первым пяти свойствам, называется простой или слабой хеш-функцией. Если кроме того выполняется шестое свойство, то такая функция называется сильной хеш-функцией. Шестое свойство защищает против класса атак, известных как атака "день рождения".

Вычисление хеш-функций

Все хеш-функции вычисляются следующим образом. Входное значение (сообщение, файл и т.п.) рассматривается как последовательность n-битовых блоков. Входное значение обрабатывается последовательно блок за блоком, и создается m-битовое значение хеш-кода. Одним из простейших примеров хеш-функции является побитовый XOR каждого блока:

$$_i = b_{i1} \oplus b_{i2} \oplus \ \ldots b_{i_k},$$

где $_i$ - i-й бит хеш-кода, $1 \leq i \leq n$, k - число n-битовых блоков входа, b_{i_j} - i-й бит в j-ом блоке. $\oplus$ – операция XOR (сложения по модулю 2).

В результате получается хеш-код длины n, известный как продольный избыточный контроль. Это эффективно при случайных сбоях для проверки целостности данных. Часто при использовании подобного продольного избыточного контроля для каждого блока выполняется однобитовый циклический сдвиг после вычисления хеш-кода. Это можно описать следующим образом:

1. Установить n-битовый хеш-код в ноль.

2.Для каждого n-битового блока данных выполнить следующие операции:

Сдвинуть циклически текущий хеш-код влево на один бит,
Выполнить операцию XOR для очередного блока и хеш-кода.

Это даст эффект "случайности" входа и уничтожит любую регулярность, которая присутствует во входных значениях.

Хотя второй вариант считается более предпочтительным для обеспечения целостности данных и предохранения от случайных сбоев, он не может использоваться для обнаружения преднамеренных модификаций передаваемых сообщений. Зная сообщение, атакующий легко может создать новое сообщение, которое имеет тот же самый хеш-код. Поэтому, чтобы защитить хеш от подделки его дополнительно надо зашифровать, используя ключ, не известный противнику. Такой зашифрованный хеш называется электронной цифровой подписью сообщения.

6.4 Алгоритм создания электронной цифровой подписи.

Электронная цифровая подпись может быть создана на основе симметричных алгоритмов, однако гораздо удобнее ее создавать, используя асимметричные алгоритмы таких, как приведенные выше RSA, схема Эль-Гамаля и шифрование на эллиптических кривых. Схема создания ЭЦП на асимметричных алгоритмах является одной и той же для любого метода и состоит из следующих шагов:

1. Сжатия электронного документа (файла) с помощью стандартной хеш-функции,
2. Шифрование хеш-свертки с помощью метода асимметричного шифрования.

Полученная строка $DS = Enc(h(F))$ и есть электронная цифровая подпись (digital signature) для сообщения F. Она прикладывается к файлу F и пересылается получателю, который получив пакет $\langle F_1, DS\rangle$ проверяет подлинность подписи, выполняя следующие действия:

1. Расшифровывает подпись, вычисляя $Enc^{-1}(DS) = h(F)$,
2. Вычисляет хеш-значение $h(F_1)$, подставляя в качестве аргумента полученный файл F_1,

3. Проверяет условие $h(F_1) = h(F)$. Если условие выполняется, то подпись верна, т.е. файл F соответствует своей подписи и не был изменен после того, как была вычислена подпись. Потенциальный взломщик не мог подделать подпись, поскольку для вычисления подписи необходимо знание закрытого ключа отправителя. Однако проверку целостности пакета по ЭЦП может осуществить любой человек, владеющий открытым ключом отправителя.

Отметим, что открытые ключи всех пользователей должны быть доступны любому зарегистрированному пользователя для проверки полученных пакетов. Для этого в сети должен быть установлен защищенный сервер аутентификации, который предоставляет указанные данные зарегистрированным пользователям по их запросам. Также задачей этого сервера является публикация скомпрометированных или вышедших из употребления по сроку действия ключей.

Для того, чтобы обмануть получателя, потенциальный взломщик должен уметь выполнять один из следующих наборов действий:

1. Иметь доступ к закрытому ключу отправителя, перехватывать послания и заменять их своими, создавая новую подпись с использованием закрытого ключа отправителя;

2. Либо взломав сервер аутентификации, заменить открытые ключи пользователей своими ключами, перехватывать послания и заменять их своими, создавая новую подпись с использованием собственного закрытого ключа, парного тому который был установлен на скомпрометированный сервер аутентификации вместо исходного открытого ключа отправителя.

Нетрудно проверить, что все свойства ЭЦП, перечисленные на с.101, выполняются для ЭЦП, сформированной на этому алгоритму. В следующем параграфе мы рассмотрим алгоритм построения ЭЦП, разработанный египетским криптографом Т. Эль-Гамалем.

6.5 Алгоритм построения ЭЦП Эль-Гамаля

В 1985 году Т.Эль-Гамаль предложил следующую схему шифрования на основе возведения в степень по модулю большого простого числа P. Алгоритм Эль-Гамаля может использоваться для формирования электронной подписи или для шифрования данных. Он базируется на трудности вычисления дискретного логарифма.

Также, как в протоколе Диффи-Хеллмана (см. раздел 6.1), сначала генерируется пара $\langle p, g\rangle$, состоящая из большого простого числа p и генератора g поля F_p. Эти параметры являются открытыми параметрами протокола Эль-Гамаля. Далее генерируются открытый и закрытый ключи:

1. Генерация ключей пользователя:

1. Сначала генерируется закрытый ключ пользователя, как произвольное число x, $1 < x < p$,
2. Вычисляет открытый ключ по формуле $y = g^x \mod p$,

2. Шифрование сообщения M:

1. Сначала генерируется случайное число k, $1 < k < p$, взаимно-простое с $p-1$, т.е. Н.О.Д$(p-1, k) = 1$,
2. Вычисляем $a = g^k \mod p$,
3. Вычисляем $b = y^k \cdot M \bmod p$.

Пара $\langle a, b\rangle$ является шифром сообщения M.

III. Расшифровка кода $\langle a, b\rangle$:

Вычисляем исходное сообщение M по формуле:

$$M = b \cdot a^{-x} \mod p$$

Отметим, что поскольку $a^{p-1} \mod p = 1$, то чтобы не вычислять обратные элементы в поле F_p, можно использовать прямое вычисление M по формуле:

$$M = b \cdot a^{p-1-x} \mod p.$$

Пример. Выберем $p = 11$, $g = 2$, секретный ключ $x = 8$. Вычисляем $y = g^x \mod p = 3$. Начальные параметры шифрования подготовлены.

Пусть сообщение $M = 5$. Выбираем случайное число $k = 9$. Убедимся, что Н.О.Д.$(k, p-1)$= Н.О.Д.$(9, 10) = 1$.

Вычисляем пару (a, b):

$$a = g^k \mod p = 2^9 \mod 11 = 3, \quad b = y^k \cdot M \bmod p = 3^9 \cdot 5 \bmod 11 = 9.$$

Получена шифрограмма $(a, b) = (6, 9)$ для сообщения $M = 9$.

Расшифровки сообщения (a, b):

$$M = b \cdot a^{p-1-x} \mod p = M = 9 \cdot 6^{11-1-8} \mod 11 = 9 \cdot 36 \mod 11 = 5$$

Замечание. При сетевом обмене сообщениями отправитель A использует для шифрования открытый ключ получателя B, поскольку только B должен иметь возможность расшифровывать адресованные ему послания. Ответные сообщения B должен шифровать, используя открытый ключ A.

Рассмотрим далее **алгоритм Эль-Гамаля построения электронной цифровой подписи**:

Генерация параметров поля F_p и выбор закрытого x и открытого y ключей пользователя происходит как и в случае шифрования. Существенная разница состоит в том, что ЭЦП строится с использованием не открытого ключа получателя, а *закрытого ключа самого отправителя.* Итак, пусть H–хеш сообщения, на которое накладывается ЭЦП.

1. Генерируется случайное число k, $1 < k < p$, взаимно-простое с $p-1$, т.е. Н.О.Д$(p-1, k) = 1$,
2. Вычисляем $a = g^k \mod p$,
3. Вычисляем b, решая уравнение

$$H = x \cdot a + k \cdot b \mod (p-1),$$

используя расширенный алгоритм Евклида.

Пара (a, b) является подписью для сообщения с хешем h.

Проверка подписи получателем сводится к вычислению хеша $H = h(M)$ полученного сообщения M и проверке условия:

$$y^a \cdot a^b \mod p = g^H \mod p$$

Приведем доказательство того, что последнее условие действительно обеспечивает целостность послания:

$$y^a \cdot a^b \mod p = (g^x)^a \cdot (g^k)^b \mod p = g^{xa+kb} \mod p$$

В силу малой теоремы Ферма $g^{p-1} \mod p = 1$, поэтому возведение g в степень H дает тот же результат, что и возведение в степень $xa + kb$, т.к. их разность делится на $p - 1$. Утверждение доказано.

Пример. Выберем параметры поля и ключи также, как предыдущем примере:$p = 11$, $g = 2$, $x = 8$, $y = 3$. Пусть хеш сообщения $H = 5$. Вычислим подпись:

1. Вычисляем $a = g^k \mod p = 2^9 \mod 11 = 3$,
2. Вычисляем b, используя расширенный алгоритм Евклида:

$$H = x \cdot a + k \cdot b \mod (p-1), \text{ или, } 5 = 8 \cdot 6 + 9 \cdot b \mod 10 \rightarrow b = 3.$$

Подпись (a, b) равна $(6, 3)$. Выполним **проверку подписи**:

$y^a \cdot a^b \mod p = 3^6 \cdot 6^3 \mod 11 = 729 \cdot 216 \mod 11 = 10$,

$g^H \mod p = 2^5 \mod 11 = 10$. Подпись верна!

Глава 7

Эллиптические кривые и их приложения в криптографии

Хотя эллиптические кривые (Elliptic Curves) исследовались на протяжении более сотни лет, интерес к ним проявляли исключительно узкие специалисты в области теории чисел. Так было примерно до 1985 г., пока одновременно и независимо Н. Коблиц (N. Coblitz) и В. Миллер (V. Miller) не предложили использовать эллиптические кривые для построения криптосистем с открытым ключом.

После этого интерес к эллиптических кривых стал расти в геометрической прогрессии. Были найдены приложения инструмента ЭК в разных областях криптографии таких, как теория кодирования, генерация псевдослучайных последовательностей, алгоритмическая теория чисел для построения тестов на простоту и, наконец, для создания одного из самых красивых методов факторизации целых чисел (Х. Ленстра [25]).

Метод факторизации Ленстры можно рассматривать как модификацию $(p-1)$–метода Полларда (см.разд 5.4). Он является самым быстрым среди всех методов, упомянутых ранее. Как и $(p-1)$–метод Полларда, сложность этого метода определяется величиной не самого числа n, а величиной его наименьшего делителя, поэтому, даже если число n очень велико и недоступно другим алгоритмам, оно может быть проверено с помощью метода факторизации эллиптических МФЭК. Подобно $(p-1)$–методу Полларда (см.разд. 5.4), МФЭК состоит из двух стадий. Первая стадия алгоритма была разработана самим Ленстрой и имеет единственный вариант. Вторая стадия имеет несколько вариаций. Одна из них, основанная на парадоксе близнецов, была предложена Брентом. [9].

В этой главе мы рассмотрим основные свойства эллиптических кривых и их приложения в теории чисел и криптографии. Среди литературы на русском языке, относящейся к теме эллиптический кривых отметим,

в первую очередь, книгу Н. Коблица «Курс теории чисел и криптографии» [59] и две книги А. Болотова, С. Гашкова, А. Фролова и А. Часовских под названием «Элементарное введение в эллиптическую криптографию» [47] и «Алгоритмические основы эллиптической криптографии» [48]. На английском языке следует отметить, в первую очередь, 2-е издание книги Л. Вашингтона «Elliptic Curves Number Theory and Cryptography». [39] Хороший обзор по алгоритмам использования эллиптических кривых в криптографии можно найти в [14].

Начнем наше изложение с определения эллиптической кривой над конечным полем.

7.1 Определение эллиптической кривой

Определение. Пусть F_q, $q = p^k$, конечное поле характеристики $p \geq 2$. Эллиптической кривой над полем F_q называется множество точек $(x, y) \in F_q \oplus F_q$, удовлетворяющих уравнению Вейерштрассе

$$y^2 + ay + b = x^3 + cx^2 + dx + e \tag{7.1.1}$$

Кроме того, к множеству точек ЭК добавляется специальная точка, обозначаемая через ∞ и называемая точкой в бесконечности.

Если характеристика поля $p \geq 3$ (а именно этот случай для нас наиболее интересен), уравнение (7.1.1) может быть преобразовано путем замены переменных в уравнение

$$y^2 = x^3 + ax + b, \tag{7.1.2}$$

где $a, b \in F_q$, которое называется *сокращенным* уравнением Вейерштрассе.

Важными параметрами эллиптической кривой являются ее *дискриминант* Δ и *инвариант* j:

$$\Delta = -16(4a^3 + 27b^2) \qquad j = \frac{1728(4a)^3}{\delta}$$

Если $\Delta \neq 0$, то многочлен $x^3 + ax + b$, стоящий в правой части уравнения кривой, не имеет кратных корней. Такая кривая называется *неособой*. Мы будем рассматривать только неособые кривые.

На множестве точек E неособой эллиптической кривой можно определить групповую операцию суммирования $+$, с помощью которой это

множество становится аддитивной абелевой группой. Нулем этой группы является бесконечно удаленная точка ∞, а обратным элементом к к точке $P = (x, y) \in E$ будет являться точка $-P = (x, -y)$.

Сначала опишем геометрическую интерпретацию операции суммирования. Пусть $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$ – произвольные точки, и $P_3(x_3, y_3)$ обозначает сумму этих точек $P_3 = P_1 \ + \ P_2$.

Проведем через точки $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$ прямую L до пересечения с третьей точкой, которую обозначим через $P'(x'_3, y'_3)$. Такая точка обязательно найдется, т.к. пересечение произвольной прямой с кривой EC имеет либо одну, либо 3 точки пересечения.

Определим сумму трех точек $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ и $P'_3(x'_3, y'_3)$ равной нулю (бесконечно удаленной точке):

$$P_1 \ + \ P_2 \ + \ P' \ = \ \infty \tag{7.1.3}$$

Тогда $P_3 = P_1 + P_2 = -P'$, откуда $x_3 = x'_3$, $y_3 = -y'_3$. Для вычисления координат точки P_3, найдем параметры прямой $L: \ y = \lambda x + d$:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}, \ d = y_1 - \lambda x_1 \tag{7.1.4}$$

Подставляя выражение для L в уравнение (7.1.1), получим

$$x^3 + cx^2 + ax + b - (\lambda x + d)^2 = 0 \tag{7.1.5}$$

Сумма координат $x_1 + x_2 + x_3$ должна быть равна коэффициенту при x^2, взятому с противоположным знаком:

$$x_1 + x_2 + x_3 = \lambda^2 - c = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - c, \tag{7.1.6}$$

откуда получим формулу для координат суммы:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 - \\ y_3 = \lambda(x_1 - x_3) - y_1 = \lambda(2x_1 + x_2 - \lambda^2 + c) - y_1 \end{cases} \tag{7.1.7}$$

где для сокращенного уравнения (7.1.2) значение параметра c равно 0.

Если точки P_1 и P_2 совпадают, то прямая L является касательной в т.P_1 и угловой коэффициент прямой L можно найти, дифференциируя уравнение (7.1.1) по x. Общие формулы для коэффициента λ получат вид:

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, \text{ если } P_1 \neq P_2, \\ \frac{3x_1^2 + a}{2y_1} \text{ если } P_1 = P_2 \end{cases} \tag{7.1.8}$$

Формулы для координат удвоенной точки можно получить из (8.1), подставляя $x_2 = x_1,\ y_2 = y_1$:

$$\begin{cases} x_3 = \lambda^2 - 2x_1 - \\ y_3 = \lambda(x_1 - x_3) - y_1 = \lambda(3x_1 - \lambda^2 + c) - y_1 \end{cases} \tag{7.1.9}$$

где опять для сокращенного уравнения (7.1.2) значение параметра c равно 0.

Группа точек эллиптической кривой над полем F_q обозначается символом $EC(F_q)$, а ее мощность (количество элементов) символом $\#EC(F_q)$.

Известно, что группа точек эллиптической кривой либо является циклической (т.е. найдется точка $P \in EC$ такая, что все точки являются кратными этой точки), либо $E(F_q) \cong Z_{n_1} \oplus Z_{n_2}$, где $n_1|n_2$, и $n_1|q-1$. Инвариант j определяет кривую с точностью до изоморфизма: любые две кривые с одинаковым инвариантом являются изоморфными (как абелевы группы).

Пример. Пусть $E(F_q)$ - группа точек кривой $y^2 = x^3 + x + 1$ над полем F_{23}. Эта группа является циклической с генератором $P(0,1)$. Рассмотрим все кратные kP точки P :

$P(0,1)$	$2P = (6,-4)$	$3P = (3,-10)$	$4P = (-10,-7)$
$5P = (-5,3)$	$6P = (7,11)$	$7P = (11,3)$	$8P = (5,-4)$
$9P = (-4,-5)$	$10P = (12,4)$	$11P = (1,-7)$	$12P = (-6,-3)$
$13P = (9,-7)$	$14P = (4,10)$	$15P = (9,7)$	$16P = (-6,3)$
$17P = (1,7)$	$18P = (12,-4)$	$19P = (-4,5)$	$20P = (5,4)$
$21P = (11,-3)$	$22P = (7,-11)$	$23P = (-5,-3)$	$24P = (10,7)$
$25P = (3,10)$	$26P = (6,4)$	$27P = (0,-1)$	$28P = (\infty)$

Таким образом, данная кривая содержит 28 точек.

Порядок точки $A = ord(A)$ — это наименьшее натуральное число k такое, что $kA = \infty$. Поскольку по теореме Лагранжа порядок любой точки является делителем порядка группы, то порядок любой точки на кривой из нашего примера принадлежит множеству $\{1, 2, 4, 7, 14, 28\}$.

Пример. Найти сумму точек $3P = (3,-10)$ и $7P = (11,3)$.

Решение. Вычислим $\lambda = (y_2 - y_1)/(x_2 - x_1)$: $y_2 - y_1 = 3 - (-10) = 13$, $x_2 - x_1 = 11 - 3 = 8$. Поскольку $8^{-1} \equiv 3 \pmod{23}$, то $\lambda = 13/8 =$

$13 \cdot 3 \bmod 23 = 16$. Теперь, $x_3 = \lambda^2 - x_1 - x_2 = (16^2 - 3 - 11) \bmod 23 = 12$, а $y_3 = \lambda(x_1 - x_3) - y_1 = 16(3 - 12) - (-10)) \bmod 23 = 4$.

Ответ: $(3, -10) + (11, 3) = (12, 4)$.

Замечание. При выполнении удвоения точки или вычислении суммы необходимо вычисление обратного элемента в поле F_q. Эта операция выполняется с помощью обобщенного алгоритма Евклида (см.разд.4.2).

Вычисление множества точек эллиптической кривой

Чтобы найти множество точек эллиптической кривой над простым полем F_p, можно использовать следующий алгоритм:

for(int $x = 0;\ x < p;\ x++$){
  int $t = x(x^2\ +\ a)\ +\ b$;
if $((t/p) == -1)$ continue;

  printf("$(x, y) = (\%d, \%d)\ \ x,\ \pm\sqrt{x}\ )$
}

В этом цикле выражение (t/p) используется для обозначения символа Лежандра.

Вычисление кратного kQ заданной точки Q

Поскольку, арифметика эллиптических кривых не содержит прямых формул для вычисления кратного kQ для заданной точки $Q(x_1, y_1)$, то эту операцию выполняют с использованием операций сложения, вычитания и удвоения точки. Для этого надо представить число k в двоичной системе исчисления $k = b_t\, b_{t-1} \ldots b_0$, $b_i \in \{0, 1\}$, потом вычислить все точки $2Q$, $4Q$, ... , $2^t \cdot Q$ и подсчитать сумму тех точек $2^i \cdot Q$, для которых $b_i = 1$.

Пример. Пусть $k = 13$. В двоичной системе $k = 1\,1\,0\,1_2$, тогда, $13Q = 8Q + 4Q + Q$. Эту же точку можно вычислить как $16Q - 2Q - Q$.

Приведем здесь фрагмент процедуры вычисления кратного kP точки P, предполагая что процедуры удвоения Double(P) и сложения точек AddPoints(P,Q) уже определены:

```
long Mult_k(Point P, long k) {
    Point B = P;
    while (k) {
```

```
        if (k%2 == 0)
          { k/ = 2; B = Double(B);}
        else
          { k − −; B = AddPoints(B, P);}
        }
      return B;
}
```

7.2 Эллиптические кривые в проективных координатах

Нахождение суммы и кратного точек ЭК требует вычисления обратного элемента в конечном поле. Это трудоемкая операция, требующая использование обобщенного алгоритма Евклида. Можно однако избавиться от частого использования этой операции, если рассматривать уравнение эллиптической кривой в трехмерных проективных координатах. Исходные координаты называются *афинными*. Точке $P(x, y)$ в афинных координатах будет соответствовать класс эквивалентности

$$(X : Y : Z) = \{(kx,\, ky,\, k)\,|k \in F_p,\, k \neq 0\},$$

который соответствует точке в проективных координатах. Выигрыш при использовании проективных координат достигается тем, что при выполнении операций удвоения или суммирования при вычислении λ по формулам (7.1.9) мы ищем обратный элемент, а просто домножаем каждую координату (X, Y, Z) на этот знаменатель. Уравнение (7.1.2) перепишется в проективных координатах как

$$Y^2Z = X^3 + aXZ^2 + bZ^3, \tag{7.2.10}$$

Бесконечно удаленной точки ∞ будет соответствовать класс $(0, 1, 0)$ проективной плоскости. Если $P = (X, Y, Z) \neq \infty$, тогда сопоставляя точке P точку X/Z, Y/Z, получим взаимно-однозначное соответствие между точками между точками кривой в афинных координатах и классами проективной ЭК.

Для получения формул сложения и удвоения точек в проективных координатах подставим в формулы (7.1.2) вместо x_1 и y_1 выражения X_1/Z_1 и Y_1/Z_1 соответственно.

Для формул удвоения получим

$$\begin{cases} \lambda = (3x^2 + a)/2y = (3X^2 + aZ^2)/2Y_1Z_1 = A_1/B_1, \\ x_2 = \lambda^2 - 2X_1/Z_1 = A_1^2/B_1^2 - 2X_1/Z_1 \\ y_2 = \lambda(3X_1/Z_1 - \lambda^2) - Y_1/Z_1 = A_1(3X_1Y_1B_1 - A_1^2)/B_1^3 - Y_1/Z_1 \end{cases} \tag{7.2.11}$$

где $A_1 = 3X_1^2 + aZ_1^2$, $B_1 = 2Y_1Z_1$.

Общий знаменатель для координат x_2 и y_2 равен $B_1^3 = 8Y_1^3Z_1^3$, поэтому для исключения знаменателя домножим координаты x_2 и y_2 на этот множитель. Получим формулы для вычисления координат удвоенной точки в проективных координатах, не использующие вычисления обратного элемента:

$$\begin{cases} A_1 = 3X_1^2 + aZ_1^2, \quad B_1 = 2Y_1Z_1 \\ X_2 = B_1(A_1^2 - 4X_1Y_1B_1) \\ Y_2 = A_1(6X_1Y_1B_1 - A_1^2) - 4Y_1^2B_1^2 \\ Z_2 = B_1^3 \end{cases} \tag{7.2.12}$$

Выпишем последовательность операций для вычисления координат удвоенной точки и подсчитаем необходимое количество операций умножения и возведения в квадрат элементов поля. Операцию умножения будем обозначать буквой M (от англ. multiplication), а возведение в квадрат буквой S (от англ. squaring). Умножения на небольшую константу и сложения мы не учитываем, поскольку эти операции имеют линейную сложность относительно длины элементов поля в то время, как операции умножения и возведения в квадрат имеют квадратичную сложность относительно длины элементов поля. Операция возведения в квадрат выполняется быстрее, чем операция умножения, и ее сложность составляет примерно от 0,6 до 0,8 от сложности умножения.

Expression	M	S
$A_1 = 3X^2 + aZ^2$	0	2
$B_1 = 2Y_1Z_1$	1	0
$T_1 = Y_1B_1$	1	0
$T_2 = 2X_1T_1$	1	0
$T_3 = A_1^2$	0	1
$X_2 = B_1(T_3 - 2T_2)$	1	0
$T_4 = T_1^2$	0	1
$Y_2 = A_1(3T_2 - T_3) - T_4$	1	0
$Z_2 = B_1^3$	1	1
Всего	6	5

Выполним те же расчеты для суммы точек.

$$\begin{cases} \lambda = (y_2 - y_1)/(x_2 - x_1) = (Y_2Z_1 - Y_1Z_2)/(X_2Z_1 - X_1Z_2) = A/B, \\ x_3 = \lambda^2 - X_1/Z_1 - X_2/Z_2 = A^2/B^2 - (X_1Z_2 + X_2Z_1)/Z_1Z_2 \\ y_3 = A(B^2(2X_1Z_2 + X_2Z_1) - A^2Z_1Z_2)/B^3Z_1Z_2 - Y_1/Z_1 \end{cases}$$

Умножая координаты x_3 и y_3 на общий знаменатель $B_1^3Z_1Z_2$, получим формулы для суммы точек в проективных координатах:

$$\begin{cases} A = Y_2Z_1 - Y_1Z_2, \quad B = X_2Z_1 - X_1Z_2, \quad C = X_2Z_1 + X_1Z_2, \\ D = 2X_1Z_2 + X_2Z_1, \quad E = Z_1Z_2, \\ X_3 = B(A^2E - B^2C), \\ Y_3 = A(B^2D - A^2E) - Y_1Z_2B^3 \\ Z_3 = B^3E \end{cases} \tag{7.2.13}$$

Для более эффективного вычисления выполним следующее преобразование. Обозначим $T_1 = X_1Z_2$ и $T_2 = X_2Z_1$. Тогда

$$B^2C = (T_2 - T_1)^2(T_2 + T_1) = (T_2 - T_1)^3 + 2T_2(T_1 - T_2)^2 = B^3 + 2T_2B^2$$
$$B^2D = (T_2 - T_1)^2(T_2 + 2T_1) = (T_1 - T_2)^3 + 3T_2(T_1 - T_2)^2 = B^3 + 3T_2B^2$$

Тогда,

$$\begin{cases} X_3 = B(A^2E - B^3 - 2B^2T_2), \\ Y_3 = A(3B^2T_2 + B^3 - A^2E) - Y_1Z_2B^3 \\ Z_3 = B^3E \end{cases} \tag{7.2.14}$$

Выпишем последовательность операций при вычислении суммы точек и оценим количество умножений и возведений в квадрат.

Expression	M	S
$T_0 = Y_1Z_2, \quad T_1 = X_1Z_2, \quad T_2 = X_2Z_1$	3	0
$A = Y_2Z_1 - T_0, \quad B = T_2 - T_1$	1	0
$A_2 = A^2, \quad B_2 = B^2, \quad B_3 = B^3$	1	2
$T_3 = B_2T_2, \quad E = Z_1Z_2,$	2	0
$F = A_2E - B_3 - 2T_3$	1	0
$X_3 = BF$	1	0
$Y_3 = A(T_3 - F) - T_0B_3$	2	0
$Z_3 = B_3E$	1	0
Всего	12	2

Рассмотрим также важный для дальнейшего описания случай смешанного сложения, когда координаты первой точки заданы в проективных координатах, а второй – в афинных координатах.

$$P + Q = (X_1, Y_1, Z_1) + (x_2, y_2) = (X_3, Y_3, Z_3).$$

$$\begin{cases} T = x_2 Z_1, \quad A = Y_1 - y_2 Z_1, \quad B = X_1 - T, \\ A_2 = A^2, \quad B_2 = B^2, \quad B_3 = B^3 \\ T_3 = B_2 T, \quad F = A_2 Z_1 - B_3 - 2T_3, \\ X_3 = BF \\ Y_3 = A(T_3 - F) - B_3 Y_1, \\ Z_3 = B_3 Z_1 \end{cases} \tag{7.2.15}$$

Стоимость последнего сложения равна 9M+2S. Это на $21,4\%$ меньше, чем стоимость сложения в обычных проективных координатах. Отметим, что при вычислении кратного kP можно взять исходную точку в афинных координатах, а вычисления производить, используя формулы для смешанного суммирования. Такое суммирование эффективно, когда множитель k достаточно велик и операция добавления точки P при вычислении kP выполняется многократно.

Приведем здесь ссылку на базу данных формул для операций удвоения и сложения для различных представлений эллиптических кривых [5]:
http:// hyperelliptic.org/EFD

7.3 Эллиптические кривые в якобиановых проективных координатах

Ускорение операции вычисления кратного точки можно получить, используя проективные якобиановы координаты. Точка в этой системе координат точки является классом

$$(X : Y : Z) = \{(\lambda^2 x,\, \lambda^3 y,\, \lambda) \,|\, \lambda \in F_p\}, \tag{7.3.16}$$

который соответствует афинной точке $(X/Z^2,\, Y/Z^3)$. Уравнение эллиптической кривой в якобиановых координатах имеет вид

$$Y^2 = X^3 + aXZ^4 + bZ^6 \tag{7.3.17}$$

Приведем сначала формулы для удвоения и сложения точек в якобиановых координатах из презентации П.Лонги и А.Мири ([27]) (см.также [28]).

1. Формулы для удвоения точки в якобиановых координатах

$$P_2(X_2, Y_2) = 2P_1(X_1, Y_1).$$

$$\begin{cases} A = 3X_1^2 + aZ_1^4, \quad B = 2\left[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4\right], \\ X_2 = A^2 - 2B, \\ Y_2 = A(B - X_3) - 8Y_1^4 \\ Z_2 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2, \end{cases} \tag{7.3.18}$$

Стоимость операции удвоения равна 2M+8S (два умножения и 8 возведений в квадрат). Напомним, что аналогичная операция в обычных проективных координатах оценивается в 6M+5S.

2. Формулы для сложения точек в якобиановых координатах

$$P_3(X_3, Y_3) = P_1(X_1, Y_1) + P_2(X_2, Y_2).$$

$$\begin{cases} A = 2(Z_1^3 Y_2 - Z_2^3 Y_1), \quad B = Z_1^2 X_2 - Z_2^2 X_1 \\ X_3 = A^2 - 4B^2 - 8Z_2^2 X_1 B^2, \\ Y_3 = A(4Z_2^2 X_1 B^2 - X_3) - 8Z_2^3 Y_1 B^3 \\ Z_3 = 2Z_1 Z_2 B \end{cases} \tag{7.3.19}$$

Стоимость этой операции равна 11M+5S, что немного уступает сложению в обычных проективных координатах (12M+2S). Значительный выигрыш в быстродействии достигается при смешанном сложении, когда первая точка задается в якобиановых координатах, а вторая – в афинных.

3. Формулы для смешанного сложения точек в якобиановых и афинных координатах

$$P + Q = (X_1, Y_1, Z_1) + (x_2, y_2) = (X_3, Y_3, Z_3).$$

$$\begin{cases} A = Z_1^3 y_2 - Y_1, \quad B = Z_1^2 x_2 - X_1, \\ X_3 = A^2 - B^3 - 2X_1 B^2 \\ Y_3 = A(X_1 B^2 - X_3) - Y_1 B^3, \\ Z_3 = ((Z_1 + B)^2 - Z_1^2 - B^2)/2. \end{cases} \tag{7.3.20}$$

Стоимость последнего сложения равна 7M+4S. Это на $21,5\%$ меньше, чем стоимость сложения в обычных проективных координатах.

Приведем также формулы для утроения точки, которая может оказаться полезной для вычисления кратных kP для специальных значений k, например, для $k = 3^t$, $t \in Z$.

$P_3 = 3 \times P_1(X_1, Y_, Z_1)$, и стоимость операции = 10M+6S.

$$\begin{cases} A = 3X_1 + aZ^4, \quad B = 8Y_1^4, \quad C = 12X_1Y_1^2 - A^2, \quad D = BC \\ X_3 = 8Y_1^2(B - D) + X_1^2C^2, \\ Y_3 = Y_1[4(D - B)(2B - D) - C^3], \\ Z_3 = Z_1C. \end{cases} \tag{7.3.21}$$

7.4 Число точек эллиптической кривой

Одной из трудных проблем, имеющих важное значение для приложений, является проблема вычисления количества точек на эллиптической кривой. Известное неравенство Хассе (Hasse) утверждает, что

$$\#E(F_q) = q + 1 - t, \tag{7.4.22}$$

где $|t| \leq 2\sqrt{q}$.

Если выполнено условие $p \,|\, t$, то кривая называется *суперсингулярной (supersingular)*, иначе кривая называется *обыкновенной (ordinary)*. Отметим, что условие $p \,|\, t$ при $p \geq 5$ эквивалентно условию $t = 0$.

Неравенство Хассе вытекает из уравнения

$$\#E(F_q) = p^k + 1 - \sum_{x \in F_{p^k}} \chi(x^3 + ax + b), \tag{7.4.23}$$

где $\chi(z)$ – квадратичный характер в поле F_q (иными словами, $\chi(z) = 1, -1, 0$ в зависимости от того, является z квадратичным вычетом, квадратичным невычетом или равен 0). Напомним, что квадратичные вычеты можно вычислять с помощью символа Лежандра (см. раздел 4.9). Однако практически формула (7.4.23) не применима, поскольку выполнение расчетов с ее использованием занимает слишком много времени.

Из неравенства Хассе вытекает, что число точек на эллиптической кривой отличается от мощности поля $q = p^n$ самое большее на величину t меньшего порядка $O(q^{1/2})$. Однако вычисления в абелевой группе точек эллиптической кривой более громоздкие, чем в конечных полях. А это значит, что для произвольной точки G вычисление множителя k такого, что $G = kP$, где P генератор точек кривой, т.е. решение проблемы, аналогичной вычислению дискретного логарифма в конечных полях, решается более трудоемко. Поэтому на группах точек эллиптических кривых можно строить криптографические протоколы типа протокола Диффи-Хеллмана выработки общего секретного ключа, электрон-

ной цифровой подписи или шифрования информации, выбирая размерность ключа (определяемую здесь размерностью поля F_{p^k}) меньшей длины. Было подсчитано, что длина ключа в 160 бит на эллиптических кривых соответствует ключу длины 1024 бита в методе RSA (см.[72], с.132).

7.5 Алгоритм факторизации Ленстры ECF

Будем обращаться к методу Ленстры факторизации на эллиптических кривых, используя аббревиатуру ECF (Elliptic Curves Factorization). Пусть n – составное число. Этот метод имеет много общего с $(p-1)$–методом Полларда, и производительность метода Ленстры зависит только от размера наименьшего делителя n, а не от размерности n.

Рассмотрим множество $Z_n = \{0, 1, 2, ..., n-1\}$ как основное множество для координат точек эллиптической кривой $EC(Z_n)$: $y^2 = x^3 + ax + b$. В строгом математическом смысле эта кривая не будет эллиптической кривой (Ленстра назвал такую кривую *псевдокривой*), т.к Z_n не является полем, и, значит, в нем не всегда выполнимы операции нахождения обратного элемента, необходимые для нахождения суммы точек кривой. Однако Ленстра заметил, что невозможность вычисления суммы двух точек $P(x_1, y_1)$ и $Q(x_2, y_2)$ означает, что разность первых координат $x_2 - x_1$ должны равняться 0 по модулю одного из делителей n, тогда, вычисляя наибольший общий делитель Н.О.Д. $(n, x_2 - x_1)$, мы легко найдем искомый делитель.

Суть алгоритма Ленстры заключается в выборе на псевдокривой $EC(Z_n)$ произвольной базовой точки P_0 и домножении ее на всевозможные простые числа и их степени пока не получим

$$kP_0 = \infty \pmod p, \tag{7.5.24}$$

где p–один из делителей n.

Замечание 1. Поскольку, ни один из делителей n нам заранее не известен, то условие выполнения (7.5.24) невозможно проверить, поэтому признаком успешного завершения работы алгоритма является выполнение условия Н.О.Д.$(n, C) = d > 1$ при очередном вычислении коэффициента λ в операции удвоения или сложения точек при вычислении очередного кратного C точки P_0.

Замечание 2. Работа алгоритма состоит из двух стадий, назы-

ваемых этапом 1 и этапом 2 (stage-one and stage-two). На первом этапе существенную роль играет настраиваемый параметр B_1, называемый ограничителем 1 этапа (stage-one limit). По сути алгоритм Ленстры является полным аналогом $(p-1)$–алгоритма Полларда (см.разд 5.4), где операция возведения в степень простого числа p заменена операцией домножения точки ЭК на множитель p. В остальном, организация работы первого и второго этапов может быть выполнена полностью аналогично работе $(p-1)$–метода.

Описание первой стадии алгоритма

I. *Инициализация:*

1. Выберем некоторое значение B_1, например, $B_1 = 10000$.

2. Выберем случайным образом числа $x, y, a \in [0, n-1]$.

3. Вычислим $b = y^2 - x^3 - ax \mod n$ и $g =$ Н.О.Д. $(n, 4a^3 + 27b^2)$. Если $g = n$, возвращаемся к п.2. Если $1 < g < n$, тогда прекратим вычисление – делитель найден. Иначе, определим кривую $E : y^2 = x^3 + ax + b$ и базовую точку-генератор $P_0(x, y)$.

4. Присвоим изменяемому параметру $P(x, y)$ начальное значение, равное P_0.

II. *Вычисление:*

1. Для каждого простого числа $p < B_1$ найдем наибольшую степень r такую, что $p^r < B_1$. Выполним цикл for $(j = 0; j < r; j{+}{+})$ $P = p \cdot P$, в результате которого точка P домножится на p^r. Каждое умножение на p выполняется с помощью алгоритма нахождения кратного точки, описанного на с.119.

2. Продолжим вычисление до тех пор, пока не будут пройдены все простые числа, меньшие B_1, или не найдется шаг, на котором выполнится условие Н.О.Д $(n, P) = d > 1$.

Если выполнится последнее условие, то искомый делитель n найден.

Иначе, либо увеличиваем B_1 и повторяем все заново, либо переходим ко второй стадии алгоритма.

Вторая стадия алгоритма

На второй стадии алгоритма предполагается, что число точек $\#EC$ на выбранной ЭК имеет лишь один делитель q, превышающий границу 1-й стадии B_1.

1. Выберем новую границу B_2, и выпишем все простые числа из интервала $[B_1;\ B_2] : \{q_1,\ q_2,\ ...,\ q_m\}$.

2. Будем последовательно вычислять точки $q_1 \cdot P,\ q_2 \cdot P,\ q_3 \cdot P,\ ...$ пока не дойдем до границы B_2, либо не выполнится условие (7.5.24).

Как и в $(p-1)$–методе Полларда, чтобы вычислить очередную точку $q_{i+1}\,P$ достаточно прибавить к ранее вычисленной точке $q_i\,P$ точку $\delta_i\,P$, где $\delta_i = q_{i+1} - q_i$. Поскольку простые числа расположены достаточно близко друг к другу, различных точек вида $\delta_i\,P$ будем немного. Их можно вычислить заранее и расположить в некотором массиве. Тогда каждое новое вычисление $S_i = q_i\,P$ можно выполнить с помощью только одной операции сложения. Однако, чтобы не пройти точки $q_i\,P = \infty$, требуется вычислять Н.О.Д.$(n,\ Z_i)$, где Z_i есть Z–координата т.$q_i\,P$ для каждого i, а каждая операция вычисления Н.О.Д. эквивалента нескольким операциям сложения. Поэтому мы рекомендуем на 2-й стадии ввести переменную $ProdZ$, первоначально равную 1, а потом на каждом шаге i выполнять вычисление $ProdZ = ProdZ \cdot Z_i (\mathrm{mod}\, n)$, где Z_i– Z–координата точки $q_i\,P$. Тогда, проверку Н.О.Д.$(n,\ Z_i) \neq 1$ можно заменить проверкой Н.О.Д.$(n,\ ProdZ) \neq 1$ и выполнять ее, например, один раз на 100 или более сложений.

Также для ускорения сложения координаты точек $\delta_i\,P$ следует представить в афинных координатах, тогда операция сложения двух точек ЭК потребует 11 умножений элементов Z_n, а не 14, как при сложении в проективных координатах (см.формулы 7.3.20).

В наиболее простом варианте реализации второй стадии можно вычислить только одну точку $2\,P$ и прибавлять ее к точке $q_1\,P$ пока не получим требуемое условие (7.5.24).

Пример 1. Пусть требуется разложить число $n = 455\,839$. Выберем эллиптическую кривую

$$y^2 = x^3 + 5x - 5,$$

точку $P = (1, 1)$ на ней и постараемся вычислить $10!\,P$.

1. Найдем сначала $2P$. Тангенс угла наклона касательной λ в т.P равен $\lambda = (3 \cdot 2 + 5)/(2y) = 4$ и координаты $P_2 = 2P = (x_2, y_2) = (14, -53)\ (\mathrm{mod} n)$.

2. Вычислим далее, $P_3 = 3(2P) = 3P_2$. Прямой формулы для вычисления точки $3P_2$ нет, поэтому придется вычислить сначала $2P_2$, затем по-

лучить $3P_2$, суммируя точки $2P_2$ и P_2. Получим $2P_2 = (259\,851, 116\,255)$, $3P_2 = (195\,045,\ 123\,227)$.

3. Продолжая эту процедуру вычислим $4!P$, потом $5!P$ и т.д. При вычислении $8!P$ знаменатель λ станет равным 599 и вычисление Н.О.Д.$(n, 599)$ даст значение $d = 599$. Отсюда 599 является делителем n, и деля n на 599 найдем второй делитель n: $455839 = 599 \cdot 761$.

Причина, по которой процесс сошелся при вычислении $8!P$, состоит в том, что кривая $y^2 = x^3 + 5x - 5 \pmod{599}$ содержит $640 = 27 \cdot 5$ точек. Вторя кривая $y^2 = x^3 + 5x - 5 \pmod{761}$ содержит $640 = 27 \cdot 5$ точек. Число 8! делится на 640, но не делится на 777. Поэтому первым появился делитель $p = 599$.

Анализ метода Ленстры

Проведем теперь анализ метода Ленстры и оценим условия, при которых он будет успешно завершен. До сих пор все вычисления проводились по модулю числа n, однако, если координаты полученных точек вычислять по модулю p, являющегося делителем n, тогда условием успешного завершения алгоритма будет, очевидно, условие

$$kP = \infty, \text{ где } k = \prod_{p_i^{a_i} \le B_1} p_i^{a_i}, \tag{7.5.25}$$

и эллиптическая кривая $y^2 = x^3 + ax + b$ рассматривается в конечном поле F_p.

Пусть $l = \#E(F_p)$ число точек этой кривой. По неравенству Хассе $l \in [p + 1 - 2\sqrt{(p)},\ p + 1 + 2\sqrt{(p)}]$. Для любой точки $Q(x, y)$ выполняется условие $lQ = \infty$, поэтому, для того, что алгоритм Ленстры успешно завершился, необходимо, чтобы множитель k в уравнении (7.5.25) делился на порядок кривой l. Последнее условие будет выполнено, если все делители l не превышают границы B_1.

Дадим здесь определение *гладкости* целого числа (smoothness), которое будет широко использоваться в последующих разделах. Пусть B – некоторое положительное целое число. Произвольное целое число x называется B – гладким, если все делители x по модулю не превышают B. Например, $x = 2^5 \cdot 5 \cdot 13^2$ является B-гладким для любого $B \ge 13$.

Это условие гладкости является более слабым, чем требуется в алгоритме Ленстры. Для успешного завершения этого алгоритма *необходимо*, чтобы все делители числа l вида p^r, кроме последнего, были меньше

границы B_1, а наибольший делитель p^r имел степень $r = 1$ и был меньше границы B_2. Например, для $\#EC(F_p) = 2^5 \cdot 5 \cdot 13^2 \cdot 233$ границы B_1, B_2 должны удовлетворять условиям $B_1 \geq 13^2 = 169$, $B_2 \geq 233$.

Число l, любой делитель которого вида p^r, где p–простое число, меньше границы B, называется *B–гладкостепенным.* .

Отметим, что необходимая граница для степеней делителей l существенно зависит от значения $\#EC(F_p)$, которое, в свою очередь, определяется коэффициентами a и b кривой. К сожалению, нет никакого регулярного способа выбрать кривую с наименьшим значением максимальной степени делителя $\#EC(F_p)$.

Пример 2. Рассмотрим простое число $p = 1007$ и вычислим наименьшие значения границ B_1, B_2 для каждого целого числа k из интервала [1001, 1013], окружающего p. Получим:

k	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013
$\mathbf{B_1}$	7	3	1	4	5	2	1	16	1	5	1	11	1
$\mathbf{B_2}$	143	167	1003	251	67	503	1007	16	1009	101	1011	23	1013

Этот пример показывает, что факторизацию числа n, имеющего в разложении множитель $p = 1007$, может выполнить при $B_1 = B_2 = 16$, а может потребовать границы B_2, сравнимой с числом 1007, в зависимости от выбранной кривой. Отметим также, что границу B_1 в большинстве случаев можно взять очень небольшой (наибольшее значение = 16), а граница B_2 почти всегда очень большая.

Поэтому процедуру факторизации на ЭК следует *всегда* выполнять одновременно с несколькими различными кривыми. На сайте http://alpertron.com.ar/E приведены оценки для значения параметра B_1 и рекомендуемое число кривых в зависимости от размерности наименьшего делителя факторизуемого числа n:

N digits	B_1	N curves
15	2000	25
20	11000	90
25	50000	300
30	250000	700
35	10^6	1800
40	$3 \cdot 10^6$	5100
45	$11 \cdot 10^6$	10 600
50	$4,3 \cdot 10^7$	19 300
55	$1,1 \cdot 10^8$	49 000
60	$2,6 \cdot 10^8$	124 000
65	$8,5 \cdot 10^8$	210 000
70	$2,9 \cdot 10^9$	340 000

Оценка эффективности метода эллиптических кривых Ленстры

Пусть наименьший множитель числа n равен p. Тогда, время работы алгоритма Ленстры можно оценить величиной

$$exp\left(\sqrt{2} + o(1)\sqrt{\ln p \ln\ln p)}\right), \tag{7.5.26}$$

которая выполняется в случае, если граница B_1 выбрана близко к величине

$$exp\left(\sqrt{2}/2 + o(1)\sqrt{\ln p \ln\ln p)}\right).$$

Поскольку значение множителя p неизвестно, то выбор значения B_1 выполняется эмпирически, что несколько ухудшает практическую оценку сходимости метода Ленстры. Отметим, что добавление в алгоритм Ленстры второй стадии вычислений сохраняет общую асимптотическую оценку, хотя обеспечивает большой практический прирост скорости сходимости алгоритма.

7.6 Рекордные разложения метода ECF

Если сравнивать метод эллиптических кривых с другими методами факторизации, то ECF относится к классу субэкспоненциальных методов факторизации, а, значит, работает быстрее любого метода, упомянутого во второй главе.

Если сравнивать его с методом квадратичного решета QS и методом решета числового поля NFS, то все зависит от размера наименьшего

делителя числа n. Если число n выбрано по методу RSA как произведение двух простых чисел примерно одинаковой длины, то метод ЕК имеет ту же оценку, что и метод квадратичного решета, но уступает методу решета числового поля.

Однако если n имеет размерность, превышающую рекордные показателя для методов QS и NFS, (напомним, что последнее рекордное разложение чисел RSA с использованием NFS относится к числу длины 768 бит), то единственная надежда найти делитель n может выполнена только с помощью метода эллиптических кривых.

За последние годы получено много новых рекордных разложений с использованием этого метода (см. сайт http://www.loria.fr/ zimmerma/records/top50.htm Самые большие делители, содержащие 73 десятичные цифры, чисел $2^{1181}-1$, $2^{1163}-1$ и $2^{1237}-1$ были найдены в 2010 году коллективом авторов, включающим Д.Босты, А.Ленстры, Т.Кляйнъюнга и П.Монтгомери. Б.Додсон нашел в 2011 году делитель, состоящий из 69 десятичных цифр, числа $2^{1822}+1$. Делитель такой же размерности числа $3^{1443}+1$ нашел в 2010 году С.Вагстафф.

Классический алгоритм Ленстры 1987 г. завершается в своей первой стадии. Последующие улучшения этого алгоритма, выполненные Монтгомери, Брентом и др., позволили решать задачу факторизации n даже в случае, если порядок l содержит более одного множителя, превышающего B_1. Описание алгоритма Монтгомери для вычисления кратного точки ЭК можно найти в книге Болотова и др. [48], а улучшения Монтгомери к методу Ленстры в статьях [31] и [32]. В следующем параграфе мы рассмотрим метод Монтгомери по книге Крендела и Померанса "Простые числа: вычислительная перспектива" [14].

7.7 "Скрученные" кривые и метод Монтгомери

Одним из чрезвычайно полезных понятий в эллиптической факторизации является понятие *скрученной* кривой (см.[14], p.329).

Определение. Пусть $E(F)$– эллиптическая кривая над полем F, заданная уравнением Вейерштрассе

$$y^2 = x^3 + Cx^2 + Ax + B,$$

и g– ненулевой элемент F, тогда *квадратичным кручением* кривой E относительно элемента g называется эллиптическая кривая над полем

F, заданная уравнением

$$gy^2 = x^3 + Cx^2 + Ax + B \tag{7.7.27}$$

Заменой переменных $X = gx$, $Y = g^2y$, эта кривая преобразуется к кривой обычного вида

$$Y^2 = X^3 + gCX^2 + g^2AX + g^3B \tag{7.7.28}$$

Будем изучать скрученную кривую, заданную уравнением (7.7.27). Питер Монгтомери предложил рассматривать точки на скрученных кривых с пропущенной координатой Y. Рассмотрим его идею первоначально для афинных координат. Пусть $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$– точки кривой (7.7.27) такие, что $P_1 \neq \pm P_2$. Обозначим через x_+, x_- x–координаты точек $P_1 + P_2$ и $P_1 - P_2$ соответственно. Следующая теорема была доказана П. Монтгомери в ([31]) и переформулирована для скрученных координат Кренделом и Померансом ([14], с.329).

Теорема. (Монтгомери [31], p.260). Пусть эллиптическая кривая EC задана уравнением

$$gy^2 = x^3 + Cx^2 + Ax + B, \tag{7.7.29}$$

точки $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ принадлежат EC, а координаты x_+, x_- определены как раньше. Тогда при $x_1 \neq x_2$ выполняется следующая формула

$$x_+ \cdot x_- = \frac{(x_1x_2 - A)^2 - 4B(x_1 + x_2 + C)}{(x_2 - x_1)^2} \tag{7.7.30}$$

При $x_1 = x_2$ аналогичная формула имеет вид:

$$x_+ = \frac{(x_1^2 - A)^2 - 4B(2x_1 + C)}{4(x_1^3 + Cx_1^2 + Ax_1 + B)} \tag{7.7.31}$$

$$x_+ = \frac{(x_1^2 - A)^2}{4(x_1^3 + Cx_1^2 + Ax_1)}$$

Доказательство. Выпишем формулу для выражения x_+:

$$x_+ = g \cdot \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - C - x_1 - x_2 \tag{7.7.32}$$

Домножим обе части на $(x_2 - x_1)^2$ и выполним преобразования, исключая gy_1^2 и gy_2^2 с использованием формулы (7.7.29) и приводя подобные члены:

$$x_+(x_2 - x_1)^2 = g(y_2 - y_1)^2 - (C + x_1 + x_2)(x_2 - x_1)^2 =$$

$$= -2g \cdot y_1y_2 + x_1x_2(x_1 + x_2 + 2C) + x_1 + x_2) = \frac{g(x_2y_1 - x_1y_2)^2 - B(x_1^2 + x_2^2)}{x_1x_2}$$

Аналогично,

$$x_-(x_2 - x_1)^2 = \frac{g(x_2y_1 + x_1y_2)^2 - B(x_1^2 + x_2^2)}{x_1x_2}$$

Перемножая обе формулы и деля результат на $(x_2 - x_1)^4$, получим

$$x_+x_- = \frac{(x_1^2 - A)^2 - 4B(x_1 + x_2 + C))}{(x_2 - x_1)^2} \tag{7.7.33}$$

Теорема доказана.

Идея использования формул (7.7.33) состоит в том, что имея координаты точек P_1, P_2 и $P_1 - P_2$ можно вычислить координаты точки $P_1 + P_2$ быстрее, чем просто вычисляя сумму $P_1 + P_2$. При этом координату y можно не рассматривать, поскольку вычисление формулы для координаты x не содержат y.

Для того, чтобы уменьшить число операций инвертирования, надо использовать проективные координаты, однако, координату Y можно опустить, поскольку координаты X и Z могут быть вычислены, без использования Y. Общие формулы будут иметь вид:

Формулы Монтгомери в проективных координатах

1. Общий случай $\mathbf{gY^2Z = x^3 + AX^2Z + BXZ^2 + CZ^3}$

$P_1 = (X_1, Y_1, Z_1),\ P_2 = (X_2, Y_2, Z_2),\ P_1 + P_2 = (X_+, Y_+, Z_+),\ P_1 - P_2 = (X_-, Y_-, Z_-)$

Случай $\mathbf{P_1 \neq P_2}$.

$$\begin{cases} X_+ = Z_- \cdot ((X_1X_2 - AZ_1Z_2)^2 - 4B(X_1Z_2 + X_2Z_1 + CZ_1Z_2)Z_1Z_2), \\ Z_+ = Z_- \cdot (X_1Z_2 - Z_1X_2)^2. \end{cases} \tag{7.7.34}$$

Случай $\mathbf{P_1 = P_2}$.

$$\begin{cases} X_+ = (X_1^2 - AZ_1^2)^2 - 4B(2X_1 + CZ_1)Z_1^3, \\ Z_+ = 4Z_1 \cdot (X_1^3 + CX_1^2Z_1 + AX_1Z_1^2 + BZ_1^3). \end{cases} \tag{7.7.35}$$

Подсчитаем количество операций (умножений M и возведений в квадрат S в конечном поле), необходимых для вычисления суммы точек и удвоенной точки:

Сложение по Монтгомери: $\mathbf{11M + 2S}$

Удвоение по Монтгомери: **10M + 3S**

В этих формулах не учтена операции сложения и умножения на константу 4 которые имеют линейную оценку относительно длины умножаемых чисел.

2. Частные случаи формул Монгомери

B = 0 :

$$\textbf{Сложение:} \quad \begin{cases} X_+ = Z_- \cdot (X_1X_2 - AZ_1Z_2)^2, \\ Z_+ = Z_- \cdot (X_1Z_2 - Z_1X_2)^2. \end{cases} \tag{7.7.36}$$

$$\textbf{Удвоение:} \quad \begin{cases} X_+ = (X_1^2 - AZ_1^2)^2, \\ Z_+ = 4X_1Z_1 \cdot (X_1^2 + CX_1Z_1 + AZ_1^2). \end{cases} \tag{7.7.37}$$

Сложение при $B = 0$: **7M + 2S**
Удвоение при $B = 0$: **4M + 3S**

C = 0 :

$$\textbf{Сложение:} \quad \begin{cases} X_+ = Z_- \cdot ((X_1X_2 - AZ_1Z_2)^2 - 4BZ_1Z_2(X_1Z_2 + X_2Z_1)), \\ Z_+ = Z_- \cdot (X_1Z_2 - Z_1X_2)^2. \end{cases} \tag{7.7.38}$$

$$\textbf{Удвоение:} \quad \begin{cases} X_+ = (X_1^2 - AZ_1^2)^2, \\ Z_+ = 4X_1Z_1 \cdot (X_1^2 + AZ_1^2 + BZ_1^3). \end{cases} \tag{7.7.39}$$

Сложение при $C = 0$: **10M + 2S**
Удвоение при $C = 0$: **7M + 3S**

Из приведенных формул видно, что в проективных координатах сокращенные формулы Монтгомери особенно эффективны для операции удвоения при $B = 0$.

Пример метода Монтгомери

Приведем пример вычисления кратного $13P$ точки P, взятый из книги Крендела и Померанса ([14], с. 331):

$$13P = (2(2P) + (2P + P)) + 2(2P + P)$$

1. $2P = doubleh(P)$,
2. $3P = addh(2P, P, P)$,
3. $4P = doubleh(2P)$,
4. $6P = doubleh(3P)$,

5. $7P = addh(4P, 3P, P)$,
6. $13P = addh(7P, 6P, P)$.

Процедура вычисления кратного метода Монтгомери

Приведем теперь алгоритм Монтгомери для произвольных точек $P(X, Z)$ и кратного k. Предположим, что $k = (k_B\ k_{B-1}\ k_0)_2$–двоичное разложение множителя k:

```
Point Mont_Mlt_k(Point P(X,Z), long k){
//————————————————————————————————
      if(k == 0) return O; // Точка в бесконечности.
      if(k == 1) return P; // Возвращаем исходную точку P.
      if(k == 2) return doubleh(P);
//————————————————————————————————
      Point U=P, T=doubleh(P);
      for(j = B − 1; j >= 0; j − −){
       if(k_j == 1)
         { U=addh(T, U, P); T=doubleh(T);}
       else
         { T=addh(T, U, P); U=doubleh(T);}
      }
      return U;
}
//————————  End of function ————————
```

Последние улучшения в ECM методе связаны с использованием кривых Эдвардса.

7.8 Кривые Эдвардса

В некоторых специальных случаях уравнение эллиптической кривой можно преобразовать к более простому виду, позволяющему выполнять операции над точками более эффективно. Кривые такого вида рассматривались еще Гауссом и Эйлером.

Использование этих кривых для криптографии началось после опубликования в 2007 году статьи Эдвардса "A normal form for elliptic curves"[16], в которой он ввел правила сложения точек на таких кривых. Эти формулы использовали существенно меньшее число операций в конечном поле,

чем все ранее изучавшиеся представления эллиптических кривых. Даниель Берштейн и Таня Ланге разработали пакет программ $EECM - MPFQ$ для быстрой факторизации чисел средних и больших размеров (см. [4], [3], [6]) с использованием кривых Эдвардса. На сайте http://eecm.cr.yp.to есть ссылки на исходные коды этого пакета.

Определение. Кривой Эдвардса называется кривая над полем K, задаваемая уравнением

$$x^2 + y^2 = c^2(1 + dx^2y^2) \tag{7.8.40}$$

При $c = 1$ кривая Эдвардса называется *скрученной кривой Эдвардса.*

Правило сложения точек на кривой Эдвардса в афинных координатах задается формулой

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}, \ \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right) \tag{7.8.41}$$

Замена переменных $x' = x/c, \ y' = y/c$ приводит к изоморфной кривой, у которой коэффициент $c = 1$, поэтому без ограничения общности будем считать в дальнейших расчетах $c = 1$.

Проективные координаты

В проективных координатах кривая Эдвардса имеет вид

$$X^2Z^2 + Y^2Z^2 = Z^4 + dX^2Y^2 \tag{7.8.42}$$

Афинные координаты точки $P(x, y)$ связаны точки связаны в проективными $P(X : Y : Z)$ отображениями

$$(x, y) \to (x : y : 1), \qquad (X : Y : Z) \to (X/Z, Y/Z)$$

причем бесконечно удаленной точке ∞ соответствует две проективные бесконечно удаленные точки $(0 : 1 : 0)$ и $(1 : 0 : 0)$.

Закон сложения в проективных координатах перепишется в виде

$$\begin{aligned}(X_1, Y_1, Z_1) + (X_2, Y_2, Z_2) = \\ = (Z_1Z_2 \cdot (X_1Y_1 + X_2Y_2), \ Z_1Z_2 \cdot (Y_1Y_2 - X_1X_2), \ Z_1^2Z_2^2 + dX_1X_2Y_1Y_2)\end{aligned} \tag{7.8.43}$$

Последовательность формул для вычисления суммы может быть записана следующим образом:

$$\begin{array}{llll} A = Z_1Z_2, & B = A^2, & C = X_1X_2, \quad D = Y_1Y_2, \quad E = dCD, \\ F = B - E, & G = B + E, & X_3 = AF((X_1 + Y_1) \cdot (X_2 + Y_2) - C - D), \\ Y_3 = AG(D - C), & Z_3 = cFG. & \end{array}$$

Число операций составляет $10\mathbf{M}+1\mathbf{S}$. Для этого вычисления было использовано равенство

$$x_1y_2 + x_2y_1 = (x_1 + y_1)(x_2 + y_2) - x_1x_2 - y_1y_2.$$

Глава 8

Отображения Вейля и Тейта

8.1 Криптографические протоколы на эллиптических кривых

В этой главе рассмотрим наиболее известные варианты использования эллиптических кривых в криптографии.

Протокол Диффи-Хеллмана

Протокол Диффи-Хеллмана используется для генерации двумя удаленными абонентами общего секретного ключа в условиях незащищенного канала связи.

Сначала выбирается простое число $\approx 2^{160}$ и параметры a и b эллиптической кривой. Этим задается эллиптическая кривая $E_p(a,b)$. Затем на $E_p(a,b)$ выбирается генерирующая точка $G=(x_1,y_1)$. При выборе т. G важно, чтобы порядок т.G $ord(G)=\min\{n \mid nG=\infty\}$ был большим простым числом. Точка G называется *базовой* точкой. Параметры $E_p(a,b)$ и координаты базовой точки криптосистемы являются открытыми параметрами, известными всем участникам. Обмен ключами между пользователями Alice и Bob (кратко, A и B) производится по следующей схеме:

1. Участник A выбирает целое число $k_A < n$. Это число является закрытым ключом участника A. Затем участник A вычисляет открытый ключ $P_A = k_A G$, который представляет собой некоторую точку на $E_p(a,b)$.
2. Точно так же участник B выбирает закрытый ключ k_B и вычисляет открытый ключ – точку на кривой $P_B = k_B G$.
3. Участники обмениваются своими открытыми ключами (координатами точек P_A и P_B), после чего вычисляют общую точку Q эллиптической кривой по следующей схеме:

Участник A вычисляет $Q = k_A \cdot P_B = k_A k_B G$, а участник B находит ключ по формуле $Q = k_B \cdot P_A = k_A k_B G$. Отметим, что поскольку точка на ЭК имеет две координаты, то можно в качестве секретного ключа взять либо только координату x точки Q, либо только координату y, либо сумму координат $x + y$.

Возможный противник, зная известные параметры k_A, k_B и G, не сможет вычислить значение общего ключа, т.к. для этого ему надо решить задачу дискретного логарифмирования на эллиптической кривой (т.е. найти кратное k по координатам т. kG и G).

Протокол Диффи-Хеллмана для трех и более участников

Идея алгоритма Диффи-Хеллмана легко может быть обобщена на несколько участников. Приведем пример для случая $n = 3$:

1. Каждый из участников A, B, и C вырабатывает секретный ключ k_A, k_B и k_C соответственно и вычисляет точки $P_A = k_A G$, $P_B = k_B G$ и $P_C = k_C G$, которые пересылает по циклу A к B, B к C, C к A.

2. Далее, участники A, B, и C вычисляют точки $Q_A = k_A P_C$, $Q_B = k_B P_A$, $Q_C = k_C P_B$ соответственно и пересылают по тому же циклу.

3. На последнем шаге вычисляется общая точка $R = k_A k_B k_C G$, домножением точки, полученной на предыдущем шаге, на соответствующий секретный ключ:

$$R = k_A k_B k_C G = k_a Q_C = k_B Q_A = k_C Q_B$$

Замечание 1. Выполнение протокола Диффи-Хеллмана для двух участников требует два обмена данными, для трех – уже шести обменов, а для произвольного n – $n!$ обменов данными, что является слишком большим числом при сравнительно небольших n. Частично эта проблема может быть решена путем использования билинейных или полилинейных отображений типа преобразований Вейля и Тейта, описываемых в следующей главе.

Замечание 2. Для суперсингулярных кривых в 1993 году был найден алгоритм Менезеса, Окатамо и Ванстоуна (так называемая MOV–атака) ([29]), основанный на преобразовании Вейля-Тейта, позволяющий свести задачу дискретного логарифмирования на ЭК (ДЛЭК) к задаче дискретного логарифмирования в конечном поле (ДЛКР), где эта задача может быть решена намного эффективнее. Поэтому суперсингулярные кривые перестали использоваться в протоколах построения электронной

цифровой подписи ЭЦП и шифрования. Однако в 2000 году А. Джоукс ([23]) нашел замечательные применения преобразованию Вейля-Тейта в криптографии, и на сегодняшний день эта тематика является одной из самых популярных а криптографии. Мы рассмотрим алгоритм Менезеса, Окатамо и Ванстоуна в разделе 8.

Шифрование сообщений с использованием эллиптических кривых

Рассмотрим самый простой подход к шифрованию/дешифрованию секретных сообщений с использованием эллиптических кривых. Задача состоит в том, чтобы зашифровать сообщение , которое может быть представлено в виде точки на эллиптической кривой $P_m(x, y)$. Как и в случае обмена ключом, в системе шифрования/дешифрования в качестве параметров рассматривается эллиптическая кривая $E_p(a, b)$ и базовая точка G на ней. Участник B выбирает закрытый ключ n_B, представляющий собой целое число от 2 до n, где n–порядок точки G и вычисляет открытый ключ $P_B = n_B \cdot G$, являющийся точкой на кривой. Участник А выбирает случайное целое положительное число k и вычисляет зашифрованное сообщение C_m, являющееся парой точек на эллиптической кривой:

$$Cm = \{k \cdot G, P_m + k \cdot P_B\}.$$

Чтобы дешифровать сообщение, участник В вычитает из второй точки произведение первой точки на свой закрытый ключ:

$$P_m + k \cdot P_B - nB \cdot (k \cdot G) = P_m + k \cdot (nB \cdot G) - nB \cdot (k \cdot G) = P_m.$$

Участник А зашифровал сообщение P_m добавлением к нему kP_B. Никто не знает значения k, поэтому, хотя P_B и является открытым ключом, никто не знает $k \cdot P_B$. Противнику для восстановления сообщения придется вычислить k, зная G и $k \cdot G$. Сделать это будет нелегко, т.к. надо вычислить дискретный логарифм. Получатель также не знает k, но ему в качестве подсказки посылается $k \cdot G$. Умножив $k \cdot G$ на свой закрытый ключ, получатель получит значение, которое было добавлено отправителем к незашифрованному сообщению. Тем самым получатель, не зная k, но имея свой закрытый ключ, может восстановить незашифрованное сообщение.

Замечание. Здесь надо еще сказать, как кодировать текстовое сообщение точками кривой. Для этого сообщение разбивается на отдельные символы, и каждый символ заменяется его числовым кодом. Можно использовать какую-нибудь стандартную кодировку типа ASCII или

WIN1251 либо просто перенумеровать все буквы последовательными цифрами.

Далее надо каждому коду сопоставить какую-нибудь точку на кривой. Самый простой вариант – это составить таблицу, сопоставив символу с кодом k координату x точки kG. Однако, лишь половина значений x является,в среднем, координатами точки ЭК. Поэтому, можно выбрать какое-нибудь натуральное число $k \geq 2$ и выбрать кодом для x любую точку ЭК $P(u, v)$, для которой $x = \lfloor u/k \rfloor$ (таких точек будет, в среднем, $\lfloor k/2 \rfloor$).

Тогда, зная координаты (u, v) любой из возможных точек P, можно восстановить x, вычисляя целую часть от u/k.

Другой вариант состоит в том, чтобы использовать эллиптические кривые специального вида, например, кривые вида $y^2 = x^3 + a \pmod p$, где $p \equiv 2(\bmod\ 3)$. Для таких кривых полином $z = x^3 + a(\bmod\ p)$ является перестановкой, поэтому каждый элемент $0 \leq u < p$ имеет кубический корень в поле F_p. Тогда, кодовую точку для числа y можно взять равной точке $P_y(x, y)$, имеющей y в качестве своей ординаты, а x, найденном из уравнения $x^3 = y^2 - a$.

Построение электронной цифровой подписи с использованием ЭК

Алгоритм ECDSA (Elliptic Curve Digest Signature Algorithm) принят в качестве стандартов ANSI X9F1 и IEEE P1363. Создание ключей:

1. Выбирается эллиптическая кривая $E_p(a, b)$. Число точек на ней должно делиться на большое простое число n.
2. Выбирается базовая точка $G \in E_p(a, b)$ порядка n, $n \cdot G = \infty$.
3. Выбирается случайное число $d \in (1, n)$.
4. Вычисляется $Q = d \cdot G$.
5. Закрытым ключом является d, открытым ключом - кортеж $< a, b, G, n, Q >$.

Создание подписи:

1. Выбирается случайное число $k \in (1, n)$.
2. Вычисляется $k \cdot G = (x_1,\ y_1)$ и $r = x_1 \pmod n$.
3. Проверяется условие $r \neq 0$, так как иначе подпись не будет зависеть от закрытого ключа. Если $r = 0$, то выбирается другое случайное число k.

4. Вычисляется $k^{-1} \pmod n$.
5. Вычисляется $s = k^{-1} \cdot ((M) + dr) \pmod n$.
6. Проверяется условие $s \neq 0$, так как в этом случае необходимого для проверки подписи числа $s^{-1} \pmod n$ не существует. Если $s = 0$, то выбирается другое случайное число k.

Подписью для сообщения M является пара чисел (r, s).

Проверка подписи:

1. Проверим, что числа r и s принадлежат диапазону чисел $(1, n)$. В противном случае результат проверки отрицательный, и подпись отвергается.
2. Вычислить $w = s^{-1} \pmod n$, и $H(M)$,
3. Вычислить $u_1 = H(M)w \pmod n$, и $u_2 = rw \pmod n$
4. Вычислить $u_1P + u_2Q = (x_0, y_0)$, $v = x_0 \pmod n$
5. Подпись верна в том и только том случае, когда $v = r$.

Дополнительную информацию об использовании эллиптических кривых в криптографии можно найти в книгах [14] и [32].

Широкое использование эллиптических кривых в криптографии основано на том свойстве, что задача дискретного логарифмирования на эллиптических кривых (задача ДЛЭК) является более трудоемкой, чем задача дискретного логарифмирования в конечных полях ДЛКП. Это позволяет использовать ключи меньшей длины по сравнению с ключами методов RSA и Эль–Гамаля (160 бит против 1024 бит), что уменьшает требования на вычислительные системы, выполняющие шифрование.

Однако в 1993 году А. Менезес, Т. Окамото и С. Вэнстоун [29] показали, что задача ДЛЭК сводится к задаче ДЛКП в некотором конечном расширении исходного поля $GF(q)$ эллиптической кривой. Идея сведения основана на спаривании Вейля (Weil's Pairing) по имени выдающегося французского математика Андре Вейля (1906–1998), известного своими трудами в области алгебраической геометрии.

Пусть задано уравнение ЭК $E: \ y^2 = x^3 + ax + b$ над полем F_q, $q = p^m$. Напомним, что алгебраическим замыканием поля K называется множество корней уравнений с коэффициентами из этого поля (обозначается $\overline{K}$).

Пусть n–целое положительное число, взаимно-простое с p. Определим множество $E[n]$ как множество точек кривой E порядка n над алгебраическим замыканием $\overline{F_q}$ исходного поля F_q , т.е. множество точек $P(x, y) \in EC(\overline{F_q})$, удовлетворяющих $nP = \infty$.

Хотя исходное поле F_q конечно, его замыкание бесконечно. Однако, множество $E[n]$ содержит конечное число элементов (можно доказать, что оно изоморфно группе $\mathbf{Z}_n \oplus \mathbf{Z}_n$) и, значит, содержится в некотором конечном расширении F_{q^k} исходного поля. Степень k называется *степенью вложения*. Эта степень может быть определена как наименьшее положительное число со свойством $n \mid (q^k - 1)$. Определим μ_n как множество корней n-й степени из 1, содержащихся в F_{q^k}.

Отображение Вейля представляет собой билинейное отображение

$$e : \ E[n] \oplus E[n] \to \mu_n, \tag{8.1.1}$$

обладающее следующими свойствами:

- (билинейность) $e(A + B, C) = e(A, C) + e(B, C)$, $e(A, B + C) = e(A, B) + e(A, C)$,
- $e(P, P) = 1$ для любого $P \in E[n]$,
- (невырожденность) $(\exists P, Q \in E[n])\ e(P, Q) \neq 1$,
- (вычислимость) $e(X, Y)$ может быть эффективно вычислено.

Если степень вложения k принимает небольшие значения (до $k = 6$), то для поиска ключа шифрования вместо решения задачи ДЛЭК можно решать более легкую задачу вычисления дискретного логарифма в конечном поле размерности q^k. Таким образом, эллиптические кривые, допускающие вложение в конечные поля с небольшой степенью k, не могут быть использованы в криптографии. Таковыми, например, являются все суперсингулярные кривые, имеющие степень вложения $k \in \{1, 2, 3, 4, 5, 6\}$.

Кривая $E = EC(GF_{p^r})$ называется *суперсингулярной*, если ее мощность $\#E = p^r + 1 - t$, и $p \mid t$.

Примером суперсингулярной кривой может служить кривая $E : y^2 = x^3 + 1 \pmod p$, если характеристика поля $p \equiv 2 \pmod 3$, тогда E содержит $p + 1$ элемент, $t = 0$ и E имеет степень вложения, равную 2.

Способ вычисления дискретного логарифма на ЭК, использующий сведение Вейля, получил называние MOV-атаки (MOV-attack) по заглавным буквам фамилий изобретателей

Многие протоколы, использующие шифрование и электронные цифровые подписи на эллиптических кривых, специально запрещают использование суперсингулярных кривых. Таким образом, суперсингулярные кривые были изъяты из криптографии.

Однако в 2002 году А.Джоукс [23] нашел неожиданное применение спариванию Вейля и суперсингулярным кривым для построения однораундового протокола выработки общего секретного ключа на основе метода Диффи-Хеллмана. Далее были найдены и другие, не менее интересные приложения такие, как, например, построение открытого ключа пользователя на основе его общеизвестных идентификационных данных таких, как, например, имя или адрес электронной почты (identity based open keys) (см. Advances in Elliptic Curve [**?**]).

8.2 Вычисление кратного точки ЭК с помощью MOV–алгоритма

Описание этого алгоритма можно найти в главе 5 книги Л. Вашингтона [39].

Пусть заданы эллиптическая кривая $EC:\ y^2 = x^3+ax+b\ (\bmod p^r)$, и точки $P, Q \in EC$ порядка n, где n–простое число, причем существует m такое, что $Q = mP$. Требуется найти множитель m. Отображение Вейля будем обозначать через $e(X, Y)$. Алгоритм вычисления m заключается в следующем:

1. Находим случайную точку $T \in EC(F_{q^k})$.

2. Находим порядок M точки T.

3. Находим d =Н.О.Д.(n, M). Если $d = 1$, то возвращаемся к п.1. Иначе, перейдем к следующему пункту. Определим, что в этом случае т.T имеет порядок n.

4. Вычислим $a = e(P, T)$ и $\ = e(Q, T)$.

5. Вычисляя дискретный логарифм в поле F_{q^k}, найдем искомый множитель m.

Отметим, что можно выполнять этот алгоритм с составным n, тогда число d может оказаться собственным делителем n и найденный множитель окажется равным $m \mod d$. В этом случае можно повторять вычисление с различными точками T_i, вычисляя $m_i = m \mod d_i$ до тех пор, пока произведение различных d_i не станет больше или равно n. После этого можно найти m с помощью китайской теоремы об остатках.

Замечание. Если речь идет о произвольной точке Q, то прежде,

чем вычислять дискретный логарифм, полезно знать, найдется ли такое m, что $Q = mP$. Эту проверку можно выполнить, используя следующее утверждение:

Теорема 8.2.1 *Для произвольной т.*$Q \in EC(F_{q^k})$ *найдется число* m *такое, что* $Q = mP$ *в том и только в том случае, если выполняются два условия:*

1. $nQ = \infty$,

2. $e(P, Q) = 1$.

8.3 Дивизоры

Построение отображения Вейля и родственного ему отображения Тейта основано на теории дивизоров (делителей) алгебраических кривых, разработанной Андре Вейлем. Приведем здесь основные сведения из этой теории. Более подробный материал можно найти в книге Л. Вашингтона [39].

Идея понятия дивизора основано на том наблюдении, что коэффициенты любого полинома можно вычислить с точностью до ненулевого множителя, зная корни этого многочлена и их кратность. Действительно, если многочлен $P(x)$ имеет своими корнями кратности r_i элементы x_i, то

$$P(x) = a \cdot \prod (x - x_i)^{r_i}.$$

В нашем случае класс изучаемых функций состоит из дробно-рациональных функций над эллиптическими кривыми, т.е. отношений двух многочленов от двух переменных x и y, определенных на точках некоторой эллиптической кривой.

Пусть теперь $E : \ y^2 = x^3 + ax + b$–эллиптическая кривая над полем K, а $f(x, y) : \ E \to K$–дробно-рациональная функция. Если f – не константа, то существует не более конечного числа точек $P \in E$, в которых $f(P) = 0$ или $f(P) = \infty$. Точки первого вида называются *нулями функции* f, а второго – *полюсами* f.

С точностью до ненулевого множителя функцию f можно задать, перечисляя все ее нули и полюсы и задавая их кратность. Если f имеет нуль (полюс) кратности k в точке P, то f можно представить в виде произведения $f = u_P^k \cdot g$, где u_P имеет в точке P нуль (полюс) первого порядка, а $g(P) \neq 0, \neq \infty$. Функция u_P называется *униформизатором* функции f в точке P

Пример. Рассмотрим кривую $y^2 = x^3 - x$ и функцию $f(x, y) = x/y$. Перепишем f в виде

$$f(x,y) = \frac{x}{y} = \frac{xy}{y^2} = \frac{xy}{x^3 - x} = \frac{y}{x^2 - 1} = y \cdot \frac{1}{x^2 - 1}.$$

Из последнего представления видим, что точка $P(0, 0)$ является нулем 1-о порядка функции $f(x, y) = x/y$, а функция $u(x, y) = y$ ее униформизатором в точке $P(0, 0)$.

Пусть M_1–множество нулей, а M_2– множество полюсов функции $f(x, y)$. Сопоставим функции f формальное выражение

$$f(x,y) \ \sim \ \sum_{P \in M_1} r_P \, [P] - \sum_{P \in M_2} r_P \, [P], \tag{8.3.2}$$

где r_P– кратность нуля (полюса) P.

Определение 8.3.1 *Пусть* $E: \ y^2 = x^3 + ax + b$ *– эллиптическая кривая над полем* k. *Дивизором* D *над кривой* E *называется формальная сумма вида*

$$D = \sum_{P \in E} r_P \, [P],$$

в которой коэффициенты r_P *– целые числа (положительные или отрицательные) и число слагаемых с ненулевым коэффициентом* r_P *– конечно.*

Множество точек P, *для которых* $r_P \neq 0$, *называется* носителем (support) *дивизора* D *и обозначается* $supp(D)$. *Целое число* $k = \sum r_P$, $P \in supp(D)$, *называется степенью* D *и обозначатся* $deg(D)$. *Точка эллиптической кривой, равная* $\sum_{P \in E} r_P \cdot P$, *называется* суммой *дивизора* D *и обозначается* $sum(D)$.

Сумма дивизоров определяется естественным образом. Множество дивизоров эллиптической кривой образует аддитивную группу относительно операции сложения, а нулем является дивизор, у которого все коэффициенты равны 0. В группе дивизоров наиболее важную роль играют дивизоры функций, которые называются главными дивизорами (principal divisors).

Вычислим дивизор прямой $l: \ ax + by + c$, проходящей через две заданные точки $P_1(x_1, y_1)$ и $P_2(x_2, y_2)$ эллиптической кривой E. Если l не является касательной в т.P_1 и P_2, то она пересекает E и в третьей т.$P_3(x_3, y_3)$, а также в бесконечно удаленной точке ∞. В точках P_1, P_2 и

P_3 прямая l имеет нули 1 порядка, а в т. ∞ – полюс 3 порядка. Чтобы увидеть это, перепишем уравнение ЭК $y^2 = x^3 + Ax + B$ в следующем виде:

$$\left(\frac{x}{y}\right)^2 = x^{-1}\left(1 + \frac{A}{x^2} + \frac{B}{x^3}\right)^{-1}, \tag{8.3.3}$$

откуда

$$x^{-1} = \left(\frac{x}{y}\right)^2 \cdot \left(1 + \frac{A}{x^2} + \frac{B}{x^3}\right). \tag{8.3.4}$$

Из уравнения (8.3.3) следует, что x/y обращается в 0 в т.∞, а уравнение (8.3.4) показывает, что функция x/y является униформизатором x^{-1} в т.∞ и т.∞ является нулем второго порядка для x^{-1}. Значит т.∞ является полюсом 2 порядка для x. Так как $y = x \cdot (y/x)$, то т.∞ является полюсом 3 порядка для y и для функции $l = Ax + By + C$. Отсюда дивизор прямой l имеет вид

$$div(l_{P_1,P_2}) = 1[P_1] + 1[P_2] + 1[P_3] - 3[\infty]. \tag{8.3.5}$$

Проведем через т.P_3 вертикальную прямую $v = x - x_3$. Она проходит через т.$P_3(x_3, y_3)$, $-P_3(x_3, -y_3)$ и т.∞, а ее дивизор имеет вид

$$div(v_{P_3}) = 1[P_3] + 1[-P_3] - 2[\infty]. \tag{8.3.6}$$

Из формул (8.3.5) и (8.3.6) получим

$$div\left(\frac{Ax + By + C}{x - x_3}\right) = div(Ax+By+C) - div(x-x_3) = [P_1]+[P_2]-[-P_3]-[\infty].$$

Так как $P_1 + P_2 = -P_3$ на кривой E, то последнюю формулу можно переписать в виде

$$[P_1] + [P_2] = [P_1 + P_2] + [\infty] + div\left(\frac{Ax + By + C}{x - x_3}\right). \tag{8.3.7}$$

Из формул (8.3.5) и (8.3.6) можно видеть, что согласно определению 8.3.1 степени прямых l_{P_1,P_2} и v_{P_3} равны 0, а их сумма равна ∞, что является примером общего факта, выражаемого следующей теоремой:

Теорема 8.3.1 *Дивизор D эллиптической кривой E, имеющий степень 0, является дивизором некоторой функции тогда и только тогда, когда $sum(D) = \infty$.*

Пример нахождения функции по заданному дивизору

Формула (8.3.7) дает способ нахождения функции f для заданного дивизора D, удовлетворяющего теореме 8.3.1. Вычислим функцию f на ЭК $E:\ y^2 = x^3 + 4x (\mathrm{mod}\, 11)$, дивизор которой имеет вид

$$D = [(0,0)] + [(2,4)] + [(4,5)] + [(6,3)] - 4[\infty].$$

Прямая l, проходящая через т.$(0,0)$ и $(2,4)$ имеет вид $l = y - 2x$, причем т.$(2,4)$ является нулем 2 порядка, откуда

$$div(y - 2x) = [(0,0)] + 2[(2,4)] - 3[\infty].$$

Вертикальная прямая через т.$(2,4)$ имеет вид $v = x - 2$ и

$$div(x - 2) = [(2,4] + [(2,-4)] - 2[\infty].$$

Значит,

$$[(0,0)] + [(2,4)] = [(2,-4)] + [\infty] + div\left(\frac{y - 2x}{x - 2}\right).$$

Аналогично,

$$[(4,5)] + [(6,3)] = [(2,4)] + [\infty] + div\left(\frac{y + x + 2}{x - 2}\right),$$

откуда

$$D = [(2,-4)] + div\left(\frac{y - 2x}{x - 2}\right) + [(2,4)] + div\left(\frac{y + x + 2}{x - 2}\right) - 2[\infty].$$

Поскольку $(2,-4)] + (2,4)] = div(x - 2) + 2[\infty]$, то получим

$$D = div(x - 2) + div\left(\frac{y - 2x}{x - 2}\right) + div\left(\frac{y + x + 2}{x - 2}\right) =$$

$$= div\left(\frac{(y - 2x)(y + x + 2)}{x - 2}\right).$$

Если раскрыть скобки в числителе и заменить слагаемое y^2 на $x^3 + 4x$, то вынося $x - 2$ за скобки, получим $(y - 2x)(y + x + 2) = (x - 2)(x^2 - y)$, откуда

$$D = div(x^2 - y).$$

Функции от дивизоров

Отображение, задаваемое формулой (8.3.9), является групповым гомоморфизмом из аддитивной группы дивизоров в мультипликативную группу поля K, т.к.

$$f(D_1 + D_2) = f(D_1) \cdot f(D_2), \quad f(D_1 - D_2) = \frac{f(D_1)}{f(D_2)}. \tag{8.3.8}$$

Распространяя формулы (8.3.8) на произвольные дивизоры, получим формулу

$$f(\sum kP) = \prod f(P)^k. \tag{8.3.9}$$

Следующая теорема носит название закона взаимности Вейля (Weil reciprocity).

Теорема 8.3.2 *Если f и g – функции на эллиптической кривой такие, что $div(f)$ и $div(g)$ не имеют общих точек, тогда выполняется следующая формула:*

$$f(div(g)) = g(div(f)).$$

8.4 Определение отображений Вейля и Тейта

Дадим в этом разделе точные определения отображений Вейля и Тейта (Weil' and Tate' Pairings). Пусть $E: \ y^2 = x^3 + ax + b$ – эллиптическая кривая над алгебраически замкнутым полем K, n–положительное целое число и $E[n]$ – подгруппа точек кривой E порядка n:

$$E[n] = \{P \in E \mid n \cdot P = \infty\}.$$

Пусть т.$P \in E[n]$. Рассмотрим дивизор $D = n[T] - n[\infty]$. Его степень равна 0, а сумма ∞. По теореме 8.3.1 найдется функция f_P, дивизор которой равен D:

$$div(f_P) = n[P] - n[\infty]. \tag{8.4.10}$$

Будем называть функцию f_P, удовлетворяющую (8.4.10), функцией Вейля. Пусть т.$Q \in E[n]$ не принадлежит орбите т.P, т.е. не совпадает ни с каким кратным kP, $k \leq n$, точки T. Рассмотрим дивизоры

$$D_Q = [Q] - [\infty], \quad D_P = [P + R] - [R], \tag{8.4.11}$$

где R–произвольно выбранная точка.

Определение 8.4.1 *Отображение (спаривание) Вейля – это билинейное отображение*

$$e_n : \ E[n] \times E[n] \to \mu_n, \tag{8.4.12}$$

где μ_n–подгруппа по умножению корней n–й степени из 1 поля K, задаваемое следующей формулой:

$$e_n(P,Q) = \frac{f_P(D_Q)}{f_Q(D_P)} \tag{8.4.13}$$

Используя формулы (8.3.8), можно переписать (8.4.13) в виде

$$e_n(P,Q) = \frac{f_P(R) \cdot f_Q(R)}{f_P(\infty) \cdot f_Q(P+R)}. \tag{8.4.14}$$

Можно доказать, что преобразование Вейля не зависит от выбора т.R, поэтому в определении (8.4.11) в качестве R можно взять любую точку ЭК. В книге Л.Вашингтона ([39]) отображение Вейля задается обратным отображением по отношению к формуле (8.4.11), полученным при перестановке местами аргументов P и Q. Это не влияет на свойства этого преобразования. Рассмотрим пример вычисления отображения Вейля.

Пример. Пусть EC–эллиптическая кривая над полем F_7, заданная уравнением

$$EC : \ y^2 = x^3 + 2.$$

Множество $E[3]$ содержит 9 точек, и $E[3] \simeq \mathbf{Z}_3 \oplus \mathbf{Z}_3$. Вычислим $e_3((0,3),\ (5,1))$.

Определим $P = (0,3)$, $Q = (5,1)$ и $R = (6,1)$. Тогда $D_P = [(0,3)] - [\infty]$, $D_Q = [(5,1) + (6,1)] - [(6,1)] = [(3,6)] - [(6,1)]$. Также как и в предыдущем разделе, найдем функцию Вейля (8.4.10) для точек P и Q:

$$f_{(0,3)} = y - 3, \quad f_{(5,1)} = \frac{4x - y + 1}{5x - y - 1}.$$

Далее

$$f_{(0,3)}(D_Q) = \frac{f_{(0,3)}(3,6)}{f_{(0,3)}(6,1)} = \frac{6-3}{1-3} \equiv 2 \pmod 7.$$

Аналогично,

$$f_Q(D_P) = 4,$$

где учтено $f_Q(\infty) = 1$ (см. [39], с.360). Отсюда

$$e_3((5,1),\ (0,3)) = \frac{2}{4} = \equiv 4 \ (\bmod 7).$$

Отметим, что 4 является кубическим корнем из 1, т.к. $4^3 = 64 \equiv 1 \pmod 7$.

Определение отображения Тейта

Определим далее отображение Тейта. Первым аргументом преобразования Тейта по-прежнему является произвольная т.$P \in E[n]$. Обозначим через nE множество точек $\{nT \mid T \in E\}$, а через E/nE множество классов эквивалентности кривой E по множеству nE.

Определение 8.4.2 *Отображение (спаривание) Тейта – это билинейное отображение*

$$\tau_n : \ E[n] \times E/E[n] \to \mathbf{F}^*_{q^k} \times \mathbf{F}^*_{q^k} \setminus \mu_n, \tag{8.4.15}$$

где μ_n–подгруппа по умножению корней n–й степени из 1 поля F_{q^k}, задаваемое следующей формулой:

$$\tau_n(P, Q) = \frac{f_P(Q + R)}{f_P(R)}, \tag{8.4.16}$$

где $R \notin \{P, -Q, P - Q, \infty\}$.

Одним из важных отличий отображение Тейта от преобразования Вейля является то, что оно не вырождено (не равно 1) при $P = Q$. Это позволяет вычислить множитель m такой, что $Q = mP$ за одно вычисление. Действительно,

$$\tau(P, Q) = \tau(P, mP) = \tau(P, P)^m = b \ \ (\bmod \ q).$$

Чтобы найти теперь m, достаточно вычислить дискретный логарифм $\log_a b \, (\mathrm{mod} q)$, где $a = \tau(P, P)$, в поле $K = F_q$.

Отметим, что значение преобразования Тейта $\tau(P, Q)$ определяется точками P и Q не однозначно, а с точностью до множителя из группы μ_n. Чтобы получить уникальное значение, элемент $\tau(P, Q)$ возводят в степень $(q^k - 1)/n$. Обозначим эту функцию через τ^*:

$$\tau^*(P, Q) = \tau(P, Q)^{(q^k-1)/n}. \tag{8.4.17}$$

8.5 Алгоритм Миллера

Главной проблемой в вычислении преобразований Вейля и Тейта является нахождение функции f, дивизор которой совпадает с заданным

дивизором D. Пусть т.$P \in E[n]$. В этом разделе будем обозначать функцию Вейля (8.4.10) с дивизором $n[P] - n[\infty]$ через $f_{n,P}$, подчеркивая ее зависимость от порядка n т.P. Определим вспомогательные дивизоры

$$D_j = j[S+R] - j[R] - [jS] + [\infty],$$

которые удовлетворяют условиям теоремы (8.3.1). Обозначим через $f_{j,P}$ функцию, дивизор которой равен D_j. Эти функции называются функциями Миллера.

Функцию Вейля $f_{n,P}(Q)$ можно вычислить с помощью рекурсивного алгоритма Миллера, основанного на вычислении промежуточных функций Миллера $f_{j,P}(Q)$ для $j < n$ по следующей формуле:

$$f_{1,P}(Q) = 1 \text{ для любой т.} Q \in E(K),$$

$$f_{i+j,P}(Q) = f_{i,P}(Q) \cdot f_{j,P}(Q) \cdot \left. \frac{l_{i,j}}{v_{i+j}} \right|_Q, \tag{8.5.18}$$

где $l_{i,j} = Ax + By + C$– уравнение прямой, проходящей через т.iP и jP, $v_{i+j} = x - x_0$– уравнение вертикальной прямой, проходящей через т. $R = (i+j)P$.

Приведем формулы для вычисления коэффициентов A, B и C прямой $l_{P,Q}$, проходящей через т.$P(x_1, y_1)$ и $Q(x_2, y_2)$:

1. $P = Q$. Угловой коэффициент λ наклона касательной равен

$$\lambda = (3x_1^2 + a)/(2y_1) \,(\bmod p). \tag{8.5.19}$$

2. $P \neq Q$. Угловой коэффициент λ равен в этом случае

$$\lambda = (y_2 - y_1)/(x_2 - x_1) \,(\bmod p). \tag{8.5.20}$$

В обоих случаях уравнение прямой, проходящей через точку $P(x_1, y_1)$ и имеющей коэффициент наклона λ, имеет вид $y - y_1 = \lambda \cdot (x - x_1)$, откуда получим уравнение l:

$$l = y - \lambda x + (\lambda x_1 - y_1). \tag{8.5.21}$$

Последними выпишем формулы для вычисления координат суммы точек $P + Q = (x_3, y_3)$ (формулы для удвоенной точки можно получить, приравнивая $x_2 = x_1$):

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = -y_1 + \lambda(x_1 - x_3). \end{cases} \tag{8.5.22}$$

Алгоритм Миллера вычисления функции Вейля $f_{P,n}$

1. Найдем бинарное представление числа $n = (n_t \dots n_0)_2$.

2. Определим исходные значения переменной точки Z и функции f равными P и 1 соответственно.

3. Выполняем цикл по i от $i = t - 1$ до $i = 0$:

- Установим
$$f = f^2 \cdot l_{Z,Z}/v_{2Z},$$
$$Z = 2Z.$$
- Если $n_i = 1$, тогда выполним операцию сложения $P + Z$:
$$f = f^2 \cdot l_{P,Z}/v_{P+Z},$$
$$Z = P + Z.$$

4. Определим выходное значение функции Вейля $f_{P,n} = f$.

Пример 1. Дана кривая $y^2 = x^3 + 11$ над полем F_{31}. Она содержит 25 точек и изоморфна группе $\mathbf{Z}_5 \times \mathbf{Z}_5$. Эта группа порождается точками $P = (2; 9)$ и $Q = (3; 10)$, имеющими порядок $n = 5$. Степень вложения $k = 1$, т.к. $p^1 - 1 = 30$ делится на $n = 5$. Вычислим функцию Вейля $f_{5,P}$, используя алгоритм Миллера:

1. Найдем двоичное представление $n = 5 = (101)_2$, $t = 2$.

2. Положим $Z = (2; 9)$. Выполним вычисления шага 3 алгоритма Миллера при $i = t - 1 = 1$.

$\lambda = 3 \cdot 2^2/(2 \cdot 9) \bmod 31 = 2/3 \bmod 31 = 2 \cdot 21 \bmod 31 = 11.$
$l = y - \lambda x + (\lambda x_1 - y_1) = y - 11x + 11 \cdot 2 - 9 = y - 11x + 13.$
$Z = 2Z = (\lambda^2 - 2x_1;\ -y_1 - \lambda(x_2 - x_1)) = (24; 28)$
$v = x - 24 \equiv x + 7.$
$f_{2,P} = (-11x + y + 13)/(x + 7).$

Проверим условие $n_i = 1$. Т.к. $n_i = 0$, то операция сложения на i–м шаге не выполняется. Переходим к следующей итерации при $i = 0$.

3. $Z = (24, 28)$. Выполним операцию удвоения т.Z: $\lambda = 22$, $l_{2,2} = 9x + y + 4$, $2Z = (2; 22)$, $v_4 = x - 2$. Отсюда

$$f_{4,P} = f_{2,P}^2 \cdot \frac{9x + y + 4}{x - 2} = \frac{(-11x + y + 13)^2(9x + y + 4)}{(x + 7)^2(x - 2)}$$

Т.к. $n_i = 1$, то выполняем вторую часть шага 3 алгоритма Миллера. Это вычисление приводит к т.$5P = \infty$ и $l = x - 2, \quad v = 1$, откуда

$$f_{5,P} = f_{4,P} \cdot (x - 2) = \frac{(-11x + y + 13)^2(9x + y + 4)}{(x + 7)^2}$$

Вычислим значение преобразования Тейта $\tau(P, Q)$, взяв $Q = (3;\ 10)$. Для этого потребуется вспомогательная точка R. Возьмем, например, $R = Q$. Вычислим сумму $S = 2Q = (-1;\ 14)$. Вычислим функцию Вейля в точках S и R:

$f(S) = f(-1;\ 14) = 20, \quad f(Q) = f(3;\ 10) = 10. \quad \tau(P, Q) = 20/10(\ \mathrm{mod}\ 31) = 2.$

Снова вычислим значение преобразования Тейта $\tau(P, Q)$, взяв $R = 2Q$. Возьмем, например, $R = 2Q = (-1;\ 14)$. Вычислим сумму $S = 2Q + Q = (-1, 17)$. Тогда

$f(S) = f(-1;\ 17) = 23, \quad f(2Q) = 20. \quad \tau(P, Q) = 23/20 = 12(\ \mathrm{mod}\ 31) = 2.$

Мы видим, что значение преобразования Тейта зависит от выбора точки R. Для получения уникального значения необходимо полученное значение возвести в степень $(q^k - 1)/n$. В нашем примере оно равно $(31 - 1)/5 = 6$. Имеем,

$$2^6\ (\ \mathrm{mod}\ 31) = 2, \quad 12^6\ (\ \mathrm{mod}\ 31) = 2.$$

Пример 2. Даны две точки $P = (2;\ 9)$ и $xP = (24;\ 3)$. Найти кратное x на эллиптической кривой $y^2 = x^3 + 11(\ \mathrm{mod}\ 31)$ из предыдущего примера.

Решение. Определим функцию $f_{P,5}$ так же, как в предыдущем примере. Вычислим $\tau(P, P)$, взяв $R = (15,\ 10)$:

$S = P + R = (2;\ 9) + (15,\ 10) = (3,\ 10), \quad f_{P,5}(S) = 30, \quad f_{P,5}(R) = 7,$

$\tau(P, P) = 30/7 \equiv 22(\ \mathrm{mod}\ 31), \ a = \tau_{un}(P, P) = 22^6 \equiv 8(\ \mathrm{mod}\ 31).$

Вычислим $\tau(P, xP)$, взяв по-прежнему $R = (15,\ 10)$:

$S' = xP + R = (24;\ 3) + (15,\ 10) = (6,\ 14), \ f_{P,5}(S') = 29,$

$\tau(P, xP) = 29/7 \equiv 13(\ \mathrm{mod}\ 31), \ b = \tau_{un}(P, xP) = 13^6 \equiv 16\ (\ \mathrm{mod}\ 31).$

Теперь для нахождения x надо вычислить $x = \log_a b(\mathrm{mod} p) = \log_8 16\ (\ \mathrm{mod}\ 31)$. Найдем x простым перебором:

$8^2\ \mathrm{mod}\ 31 = 2, \quad 8^3\ \mathrm{mod}\ 31 = 16,$ значит $x = 3$.

8.6 "Перемешивающий"эндоморфизм эллиптической кривой

При вычислении значений преобразования Вейля и Тейта возникает вспомогательная задача вычисления по заданной точке P эллиптической кривой точки Q линейно не зависимую от P. Эту операцию удобно выполнять, используя перемешивающий изоморфизм (a distortion map) множества точек кривой , который представляет собой эндоморфизм $\phi(P)$ кривой E, переводящий точку P в точку Q, не совпадающую ни с какой кратной mP точки P (см. Verheul, [**?**]). Верхойл показал, что нетривиальные перемешивающие отображения существуют почти для всех суперсингулярных кривых. В следующей таблице мы дадим описание перемешивающих эндоморфизмов для основных классов суперсингулярных эллиптических кривых.

Поле	Кривая	Отображение	Условия	$Ord(EC)$
F_p	$y^2 = x^3 + ax$	$(x, y) \to (-x, iy)$ $i^2 = -1$	$p \equiv 3 (\mathrm{mod}\ 4)$	$p + 1$
F_p	$y^2 = x^3 + b$	$(x, y) \to (\eta x, y)$ $\eta^3 = 1,\ \eta \neq 1$	$p \equiv 2 (\mathrm{mod}\ 3)$	$p + 1$
F_{p^2}	$y^2 = x^3 + b$, $b \notin F_p$	$(x, y) \to \left(\omega \frac{x^p}{r^{(2p-1)/3}}, \frac{y^p}{r^{p-1}}\right)$, $r^2 = b,\ r \in F_{p^2}$ $\omega^3 = r,\ r \in F_{p^6}$	$p \equiv 2 (\mathrm{mod}\ 3)$	$p^2 - p + 1$

Например, кривая $E:\ y^2 = x^3 + 3x\ (\mathrm{mod}\ 11)$ принадлежит к классу, описанному в первой строке таблицы, и перемешивающий эндоморфизм имеет вид $(x, y) \to (-x, iy)$. Например, точка $P(3;\ 5) \in E$ переходит в точку $\phi(P) = (-3;\ 5i) \in E$.

Используя перемешивающий эндоморфизм ϕ, можно модифицировать преобразование Вейля, чтобы оно не было вырожденным при $P = Q$. Модифицированное преобразование Вейля $\hat{e}_n(P, Q):\ E[n] \times E[n] \to \mu_n$ определяется формулой

$$\hat{e}_n(P, Q) = e_n(P, \phi(Q)). \tag{8.6.23}$$

8.7 Приложения преобразований Вейля и Тейта

В статьях ([23], [24]) А. Джоукса приведены примеры использования преобразований Вейля и Тейта в криптографии. Рассмотрим в этом разделе эти приложения.

1. Протокол Диффи-Хеллмана для трех участников

Протокол Диффи-Хеллмана генерации общего секретного ключа для трех участников (Tripple Diffi–Hellman) был рассмотрен в предыдущей главе. Дадим его описание с использованием преобразований Вейля–Тейта.

1. Рассматривается эллиптическая кривая $EC:\ y^2 = x^3+ax+b(F_q)$ и точка P большого порядка n.

2. Каждый из участников A, B и C выбирает случайное число k_A, k_B и k_C на интервале $[2;\ n-1]$ и вычисляет точку k_AP, k_BP, k_CP соответственно, которую пересылает остальным участникам.

3. Теперь каждый участник вычисляет общий элемент конечного расширения поля F_{q^k}, где $n|q^k-1$ по формуле

$$k = \tau(P,P)^{k_Ak_Bk_c} = \tau(bP,cP)^{k_A} = \tau(aP,cP)^{k_B} = \tau(aP,bP)^{k_C}$$

Выигрышем применения преобразований Вейля–Тейта является уменьшение числа информационных обменов по сети (от 6 до 3).

2. Шифрование на основе идентификационных данных пользователей

Идея шифрования на основе идентификационных данных пользователей (Identity based encryption – IDE) была впервые выдвинута в 1984 году А.Шамиром. Она состоит в том, чтобы использовать в качестве открытого ключа пользователя не произвольные труднозапоминающиеся ключи, а естественные ключи, полученные из общеизвестных сведений о пользователе, например, его фамилии и инициалов, электронного адреса или другого известного другим пользователям идентификатора. В случае использования схемы IDE отпадает потребность в сложной системе лицензированных сертификационных центров и хранении базы данных открытых ключей на защищенном сервере.

До упомянутой статьи Джоукса было предложено несколько реализаций IDE. Например, в 2001 году на конференции Cryptography and Coding Кокс (C.Cocks) предложил схему [12], основанную на классической задаче определения, является ли заданной число квадратичным вычетом в конечном поле. Оно было слишком громоздким, т.к. для шифрования 1 бита информации требовалось два числа RSA.

Боне и Франклин in [8] предложили другое решение, основанное на суперсингулярных кривых и спариваниях. Дадим здесь описание идеи.

Пусть m–сообщение, которое требуется подписать, используя публичный идентификатор ID_A пользователя Alice, например, Alice@gmail.com.

Первая проблема состоит в том, чтобы закодировать эту строку точкой на ЭК. Решение может быть таким, как описано в разделе "Криптографические протоколы" (с. 141).

Формирование подписи

1. Сначала выбираются глобальные параметры системы: эллиптическая кривая ЭК, базовая т.P порядка n и глобальный секретный ключ s. По ним вычисляется открытая точка $Q = sP$.

2. Далее, открытому идентификатору Id_A сопоставляется точка P_A, имеющая также порядок n. Эта точка является открытым ключом A, а соответствующим закрытым ключом является точка $Q_A = sP_A$. Пользователю A не нужно знать значение s.

Для того, чтобы подписать сообщение m необходимо выполнить следующие действия:

1. Выбираем случайное число r из диапазона $[2; n-1]$.
2. Формируем криптографическую подпись

$$sg(m) = \langle rP,\ m \oplus H(\tau(P_A,\ rQ))\rangle,$$

где H–криптографическая хеш-функция.

Проверка подписи

Для проверки подписи необходимо вычислить $\tau(P_A,\ rQ)$, используя следующие соотношения:

$$\tau(Q_A,\ rP) = \tau(P_A,\ rP)^s = \tau(P_A,\ rQ)$$

Если требуется, чтобы ключ пользователя периодически обновлялся, можно вместо постоянного адреса Alice@gmail.com кодировать конкатенацию строк Alice@gmail.com || Текущий_Год.

3. Слепая подпись

Схема слепой подписи используется в тех случаях, когда подписывающее лицо не должно знать содержимое документа. Понятие слепой подписи было введено Девидом Чаумом в 1990 году в работе [11]. В этой же работе он предложит первую реализацию слепой подписи с использованием метода RSA:

Пусть n–число RSA, (e, d)–пара, состоящая из открытого и закрытого ключей подписывающего лица. Сообщение M кодируется числом $m \in [2; n-1]$. Алгоритм получения слепой подписи состоит из следующих шагов:

1. Выбираем маскирующий множитель k, равным случайному числу из диапазона $[2; n-1]$.

2. Маскируем сообщение m, домножая его на k: $m' = k^e \cdot m \mod n$.

3. Передаем m' на подпись:

$$s(m') = (m')^d \bmod n = (mk^e)^d \bmod n = km^d \bmod n$$

4. Снимаем маскирующий множитель

$$s(m) = s(m')/k.$$

Опишем теперь этот же алгоритм, используя билинейные спаривания.

Слепая подпись на эллиптических кривых

Пусть даны: эллиптическая кривая ЭК, базовая т.P порядка n на ЭК, секретный ключ s и открытый ключ $Q = sP$ подписывающего лица, точка ЭК Q_m, кодирующая сообщение m. Построение подписи выполняется следующим образом:

1. Выбираем маскирующий множитель k, равным случайному числу из диапазона $[2; n-1]$ и вычисляем kP.

2. Вычисляем точку $R = Q_m + kP$ и передаем ее на подпись.

3. Подписывающее лицо ставит подпись

$$s(R) = s \cdot R = s \cdot (Q_m + kP) = sQ_m + skP = s(Q_m) + kQ$$

4. Снимаем маскирующий множитель с подписи

$$s(Q_m) = s(R) - kQ$$

В работе Александры Болдыревой [7] приведены другие примеры построения подписей (мультиподписей, коротких подписей), основанных на трудноразрешимости *вычислительной проблемы Диффи-Хеллмана–ВПДХ*. Напомним, что ВПДХ – это проблема вычисления в конечной группе $\langle G, g\rangle$ с порождающей g по заданным элементам g^a, g^b элемента g^{ab}.

Глава 9

Генерация криптографических ключей

9.1 Введение

Генерация криптографического ключа является составной частью любой системы защиты информации. Иногда к этому ключу предъявляются дополнительные требования, например,представление в виде произведения двух простых чисел, однако, часто единственным требованием является случайность сгенерированной последовательности. На практике, вместо случайных чисел часто используют псевдослучайные числа. Способам создания генераторов псевдослучайных чисел посвящены многочисленные работы. В первой части данной главы кратко остановимся на методах порождения таких чисел и проблемах, связанных с возможностью восстановления последовательности по ее части. В основном речь пойдет об устройствах на основе сдвиговых регистрах с обратными связями. С математической точки зрения это теория линейный последовательностных машин (ЛПМ). Теория таких устройств нашла освещение в многочисленных статьях и монографиях, в связи с чем рассмотрение указанных устройств носит обзорный характер. Теория генераторов, использующих физические датчики, также разработана достаточно хорошо, но она гораздо меньше представлена в учебной литературе. Лишь недавно опубликована монография [77], где рассматриваются различные способы реализации таких датчиков. Однако, там основное внимание уделяется технической стороне вопроса. Теории физических датчиков посвящена большая часть данной главы. Во второй части рассматривается гибридный подход, когда используется физический датчик случайных чисел и преобразующая его ЛПМ. Такие датчики принято называть комбинированными генераторами псевдослучайных чисел (КГПСЧ) [77]. При этом

не уточняется тип физического источника случайных величин.

В третьей части рассматривается конкретный тип источника случайных чисел, основанный на работе комбинационной схемы в режиме "дребезжания"(jittering). Четвертая часть посвящена возможности применения аппаратуры, работающей в трехзначной логике, для создания датчиков случайных чисел.

При работе с объектами линейной алгебры в данной главе приняты следующие обозначения. Вещественные или комплексные числа обозначаются строчными латинскими буквами, векторы обозначаются жирным латинским шрифтом либо буквами греческого алфавита. Например, вектор-строка, имеющий n компонентов, записывается в виде $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$, либо как $\xi = \langle a_1, \ldots, a_n \rangle$ Матрицы обозначаются прописными латинскими буквами. Элемент матрицы A, стоящий в строке с номером i и столбце с номером j, обозначается символом $A[i|j]$. Строка с номером i и столбец с номером j обозначается символами $A[i|*]$ и $A[*|j]$ соответственно. Единичная матрица обозначается символом I, а диагональная матрица с элементами $d_1, \ldots, d_n$ обозначается как $diag(d_1, \ldots, d_n)$. Последовательности рассматриваются, как матрицы. Символ $A[p]$ означает элемент последовательности A с индексом p. Часть вычислений производится с вещественными и комплексными числами, а часть — в поле $GF(2)$ вычетов по модулю 2. Умножение во всех полях обозначается одинаково, но для сложения в поле $GF(2)$ будет применяться символ $\oplus$.

От читателя требуются знания в объеме стандартных курсов линейной алгебры, теории вероятностей и дискретной математики, преподаваемых на математических и инженерных факультетах. Все понятия, выходящие за пределы этих курсов, объясняются, и даются необходимые ссылки. Все необходимые вычисления в приводимых примерах осуществляются с помощью открытого математического пакета SciLab [36].

9.2 Матрицы с неотрицательными элементами

В процессе исследования свойств генераторов будет существенно использоваться теория матриц с неотрицательными элементами. Все необходимые сведения вместе с доказательствами можно найти в [53],[71] и [44]. Здесь будут приведены основные факты, относящиеся к этим объектам.

Определение 1 *Матрица A называется неотрицательной (положительной), если все элементы этой матрицы неотрицательные (положи-*

тельные) числа. Эти условия записываются, как $A \geq 0$ *и* $A > 0$ *соответственно.*

Определение 2 *Неотрицательная квадратная матрица* A *называется разложимой, если существует матрица перестановки* P *такая, что*

$$P^T \cdot A \cdot P = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix},$$

в противном случае матрица называется неразложимой

Из определения следует, что если матрица A является разложимой, то тоже самое справедливо и для любой ее степени, и матрица $I + A$ также будет разложимой. С другой стороны, справедливо

Предложение 1 *Если матрица* A *порядка* n *является неразложимой матрицей, то*

$$(I + A)^{n-1} > 0 \tag{9.2.1}$$

Другими словами, условие (9.2.1) является необходимым и достаточным для неразложимости неотрицательной матрицы

Другой признак неразложимости дает

Предложение 2 *Неотрицательная матрица* A *будет неразложимой тогда и только тогда, когда для любой пары индексов* i, j *существует натуральное число* q *свое для каждой пары такое, что* $A^q[i|j] > 0$

Теорема 1 *Если* A *неразложимая матрица, то существует вещественное характеристическое число* r *этой матрицы такое, что имеет место неравенство* $|c| \leq r$ *для любого другого характеристического числа* c *этой матрицы, число* r *является простым корнем характеристического уравнения, и ему принадлежит собственный вектор* α *матрицы* A *с положительными элементами.*

Определение 3 *Матрица* $A \geq 0$ *называется стохастической, если*

$$(\forall i) \sum_k A[i|k] = 1$$

и дважды стохастической, если стохастическими являются матрицы A *и* A^T

Для неразложимой стохастической матрицы $r = 1$.

9.3 Генерация псевдослучайных чисел

При изложении данного параграфа будем следовать книге [43]. С математической точки зрения генераторы псевдослучайных чисел работают следующим образом. Имеется автономный конечный автомат $\Xi = (Y, Q, \delta, \lambda, q_0)$. Здесь Y – выходной алфавит, Q – конечное множество состояний, $\delta : Q \to Q$ – функция переходов автомата, $\lambda : Q \to Y$ – функция выхода, q_0 – начальное состояние. Перед началом работы Ξ устанавливается в начальное состояние. Время считается дискретным. Если в некоторый момент времени n автомат находится в состоянии q_n, то в этот момент времени на выходе автомата появляется сигнал $y_n = \lambda(q_n)$, а в момент времени $n+1$ автомат окажется в состоянии $q_{n+1} = \delta(q_n)$. Часто для описания автомата вместо функции δ используют диаграмму переходов. Это направленный граф, вершинами которого являются состояния автомата, а дуга из вершины q в вершину q' присутствует тогда и только тогда, когда $\delta(q) = q'$. Очевидно, что из каждой вершины выходит только одна дуга, но возможно, что в некоторую вершину входят несколько дуг. Вся диаграмма переходов распадается на несколько связных компонент. Если в каждую вершины компоненты входит только одна дуга, то такая компонента называется циклической, или кольцевой.

В силу конечности числа состояний автомата, какое-то состояние должно повториться — для некоторых целых n, p будет выполнено равенство $q_n = q_{n+p}$. Это означает, что для любого натурального числа k выполнено равенство $y_{n+k} = y_{n+p+k}$. В дальнейшем последовательность на выходе становится периодической с периодом p. Указанное свойство является принципиальным недостатком любого генератора псевдослучайных чисел, поэтому при проектировании устройства стараются сделать параметр p как можно большим. Достоинством данного метода является простота его реализации на компьютере и возможность повторить сгенерированную последовательность столько раз, сколько надо. Все математические пакеты содержат подобного рода генераторы, реализующие один из алгоритмов. Как правило, генератор должен создавать равномерное распределение на интервале [0,1], поскольку из такого генератора можно получить теоретически любое другое распределение. Для получения равномерности достаточно потребовать выполнения следующего условия: диаграмма переходов является кольцом большой длины. Действительно, если $|Q| = r$, то перенумеруем произвольным образом все состояния числами от 1 до r и положим $\lambda(q_i) = i/r, \quad i = 1, \ldots r$. В этом

случае в результате прохода по кольцу получим на выходе каждое из чисел i/r ровно один раз. Однако, одной равномерности не достаточно, требуется чтобы числа на выходе были "независимыми". Формально ни о какой независимости не может быть и речи, поскольку полученная последовательность является детерминированной. В этой связи под независимостью понимают прохождение полученной последовательности через несколько статистических тестов. Рассмотрим два наиболее распространенных и простых в применении теста.

9.3.1 Автокорреляционная функция

Напомним известный из теории вероятности факт. Имеются две независимые случайные величины ξ и η с нулевым средним. В этом случае $E(\xi\eta) = 0$. Пусть имеется генератор псевдослучайных величин $x_n = f(n)$. Полученные числа рассматривают как реализацию некоторого стационарного случайного процесса с нулевым средним. Функция $R(p) = E(x_n x_{n+p})$ называется автокорреляционной функцией процесса. Если случайные величины x_n и x_{n+p} независимы, то $R(p) = 0, p > 0$. При изучении свойств конкретного устройства либо подсчитывают теоретическое значение функции

$$R_r(p) = \frac{1}{r}\sum_{k=1}^{r} x_k x_{k+p} \tag{9.3.2}$$

(напомним, что в (9.3.2) последовательность x_n является периодической с периодом r) либо выбирают отрезок длины m сгенерированной последовательности и подсчитывают

$$R_m(p) = \frac{1}{m}\sum_{k=1}^{m} x_k x_{k+p}, \tag{9.3.3}$$

дополняя, если необходимо, полученную последовательность нулями. Если условие $R_m(p) \to 0, \quad m \to \infty$ не выполнено, то тест на независимость не проходит, хотя близость этой величины к нулю не обеспечивает независимость. На практике проверяют отсутствие отдельных пиков в графике функции $R_m(p)$. Наличие пика в точке p_0 означает зависимость между значениями с шагом p_0. Отметим, что вычисления в (9.3.3) производятся с помощью быстрого преобразования Фурье (FFT) (например, см. [75]). С этой целью берется вектор длины $2m$

$$\langle x_1, \ldots, x_m, 0, \ldots, 0\rangle.$$

вычисляется преобразование Фурье $\mathbf{F}$ от этого вектора, которое само является вектором длины $2m$, поэлементно перемножаются компоненты векторов $\mathbf{F}$ и $\bar{\mathbf{F}}$ – состоящего из сопряженных чисел – и берется обратное преобразование Фурье от этого вектора. Последовательность, образованную из компонентов вектора обозначим через G. В результате

$$R_m(p) = \frac{1}{m-p} G[p], \quad p = 0, \ldots, m/2 \tag{9.3.4}$$

Появление коэффициента в (9.3.4) обусловлено тем, что из-за наличия нулей реальное количество слагаемых в (9.3.4) уменьшается с ростом p. Применим этот метод для проверки качества стандартного датчика псевдослучайных чисел, реализованного в пакете SciLab. В консоли введем следующий скрипт (номера операторов вводить не надо)

```
N=200; //1
x1=rand(1,N,'normal');//2
 x2=[x1,zeros(1,N)];//3
fx2=fft(x2);//4
cfx2=conj(fx2);//5
prd=fx2.*cfx2;//6
crr=real(ifft(prd));//7
K=N:-1:1; //8
auto=crr(1:N)./K;//9
plot(auto)//10
```

Поясним смысл операторов.

1. выбор длины исследуемой последовательности
2. генерация N псевдослучайных чисел, имеющих нормальное распределение с нулевым средним и единичной дисперсией
3. добавление N нулей
4. вычисление преобразования Фурье
5. нахождение сопряженного массива чисел
6. поэлементное умножение
7. вычисление обратного преобразования Фурье и выделение вещественной части; формально, должно получиться вещественное число, но возможно появление комплексной части из-за ошибок округления

8. создание массива чисел от N до 1

9. поэлементное деление двух массивов

10. график получившегося массива представлен на Рис.9.1

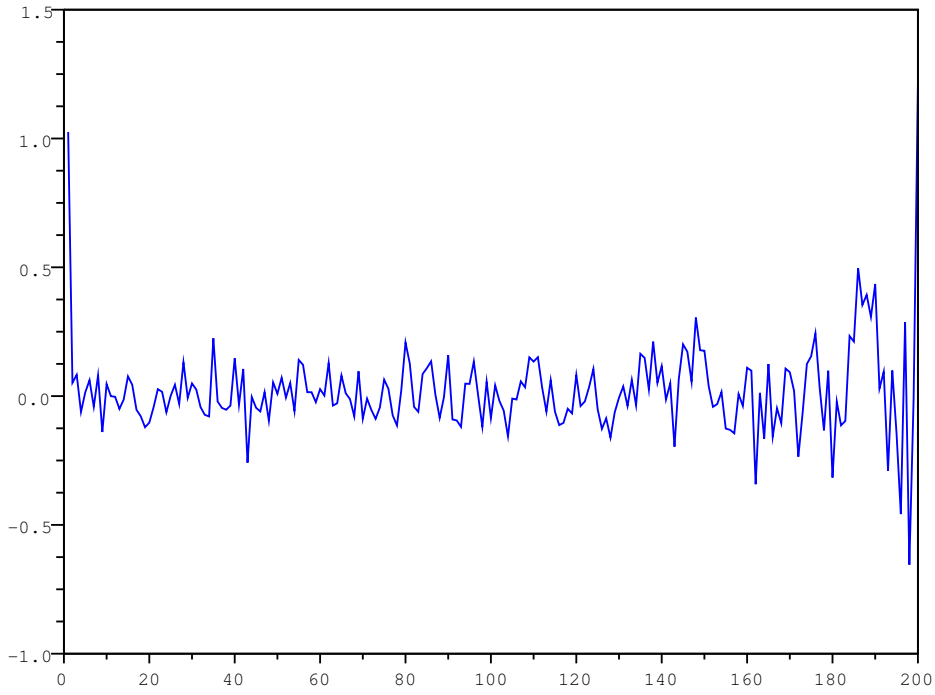


Рис. 9.1: Автокорреляционная функция

Обратим внимание на на большую разницу в значениях $R(0)$ и $R(p)\, p \neq 0$. Исключение составляют числа в конце массива. Это связано с тем, что указанные значения получаются усреднением по малому количеству слагаемых. Оценка качества в данном случае осуществляется визуально. Следующий способ позволяет дать численную оценку качества генератора.

9.3.2 Критерий χ^2

Значительная часть критериев независимости случайных величин основана на распределении χ^2_n. Напомним, что это распределение имеет ве-

личина

$$\xi_n = \sum_{k=1}^{n} \eta_k^2,$$

где η_k независимые случайные величины с нормальным распределением нулевым средним и единичной дисперсией. Различные критерии основаны на следующем факте [62]. Пусть имеются n независимых случайных величин μ_k, имеющих одно и то же распределение. Разобьем область значений случайных величин на q частей: $S_1, S_2, \ldots, S_q$. Обозначим через p_i вероятность $p_i = P(\mu \in S_i)$, а через n_i количество величин $\mu_k \in S_i$, действительно попавших в это множество в результате эксперимента. Тогда значение

$$v = \sum_{i=1}^{q} \frac{(n_i - np_i)^2}{np_i} \tag{9.3.5}$$

имеет распределение близкое к χ^2_{q-1} для больших n. Выбирается уровень значимости, например, 5 процентов. По таблицам находят значение z, для которого выполнено равенство $P(\xi_{q-1} > z) = 0.05$. Если окажется, что выполнено неравенство $v > z$, гипотезу о равномерности распределения величин μ_k отвергают, в противном случае ее принимают.

Покажем, как общая схема применяется для проверки качества генератора. Выбирается начальное состояние генератора и длина последовательности. Последняя должна быть значительно меньше общей длины кольцевой компоненты, которой принадлежит начальное состояние. Предположим, что все порожденные значения x_k находятся в интервале [0,1]. В этом случае этот интервал разбивается произвольным образом на q частей $S_i, i = 1, \ldots, q$, вычисляется значение согласно (9.3.5) и проверяется справедливость гипотезы. Эксперимент повторяется многократно при различных начальных состояниях.

Более тонкий способ проверки выглядит следующим образом [55]. Выбирается натуральное число w и строятся векторы вида

$$\mathbf{b_1} = \langle x_1, \ldots, x_w \rangle, \mathbf{b_2} = \langle x_{w+1}, \ldots, x_{2w} \rangle, \ldots$$

w-мерный куб разбивается на q частей $S_i, i = 1, \ldots, q$ и снова подсчитывается значение (9.3.5) для величин $\mathbf{b_k}$, после чего справедливость гипотезы о равномерности оценивается прежним способом. Следует отметить, что создание генератора, проходящего указанный тест для больших w является серьезной проблемой.

В качестве примера осуществим проверку того же датчика, но используя изложенный критерий.

В SciLab в консоли выпишем следующий скрипт

```
N=200;//1
x=rand(N,4);//2
buff=zeros(1,16);//3
pows=[1,2,4,8];//4
for k=1:N //5
    flags=zeros(1,4); //6
    str=x(k,:);//7
    for i=1:4//8
        if(str(i)>0.5),flags(i)=1;end//9
    end//10
    numb=1;//11
    for i=1:4  //12
        numb=numb+flags(i)*pows(i);//13
    end//14
    buff(numb)=buff(numb)+1;//15
end//16
```

Операторы 1-2 порождают матрицу x размера $N \times 4$, составленную из псевдослучайных чисел с равномерным распределением на [0,1]. Четырехмерный единичный куб разделим на 16 равных частей. Вектор длины 4 $\eta = \langle a_1, a_2, a_3, a_4 \rangle$ попадает в ту или иную часть, в зависимости от того, справедливо ли неравенство $a_i < 0.5$, $i = 1, 2, 3, 4$. Оператор 3 создает массив $buff$, в котором элемент $buff(j)$, $j = 1, \ldots, 16$ показывает, сколько векторов из выборки попало в часть с номером j. Оператор 6 создает массив, в котором будет записан в двоичной форме номер области, в которую попал очередной вектор. Оператор 7 выделяет очередную строку, а операторы 8-10 формируют двоичный номер. Операторы 11-14 переводят двоичный номер в обычное число. Единица добавляется в номер, поскольку все индексы начинаются с 1.

После того, как будет сформирован $buff$, надо применить критерий χ^2 для проверки гипотезы о равномерности распределения векторов по областям. В данном случае попадание в каждую область равно $p_i = 1/16$. Чтобы применить формулу (9.3.5), надо выполнить следующий скрипт

```
buff1=buff-N/16;//1
v=sum(buff1.*buff1)/(N/16); //2
```

Оператор 1 вычитает из всех элементов $buff$ одно и то же число, а оператор 2 завершает подсчет по формуле (9.3.5).

Полученное число v надо сравнить с порогом, определенным уровнем значимости. Для этого выбираем этот уровень, например 0.05, и подсчитываем порог с помощью встроенной функции SciLab. Вводим

```
val=cdfchi("X",15,0.95,0.05)
```

Здесь 15=16-1 — число степеней свободы, два оставшиеся числа определяют уровень значимости (их сумма всегда 1, но порядок следования аргументов существенен). Если окажется, что $v > val$, гипотеза отвергается. Конечный результат зависит от начального состояния генератора псевдослучайных чисел. В проведенном эксперименте получилось значение $v = 20$, в то время как $val = 24.99$. Это означает, что гипотеза о равномерности попадания векторов в каждую область надо принять. Интересно провести проверку для векторов, длина которых порядка 20-30, но для этого нужен достаточно мощный компьютер.

9.4 Способы построения генераторов

Рассмотрим два наиболее распространенных способа построения генераторов псевдослучайных чисел. Первый из них ориентирован на применение компьютера, а второй — на аппаратную реализацию.

9.4.1 Мультипликативный датчик

Мультипликативный датчик выглядит следующим образом [58]. Выбираются натуральные числа q, a, n, x_0. Множество состояний автомата состоит из целых чисел $0, 1, \ldots, q-1$. Состояния меняются согласно формуле

$$x_k = (a * x_{k-1} + n) \pmod q \quad k = 1, 2, \ldots$$

Функция выходов $\lambda(x) = x/(q-1)$. В результате сгенерированные числа принадлежат интервалу [0,1]. Рекомендации по выбору параметров q, a, n представлены в [58]. Следует отметить, что эти рекомендации учитывают как статистические свойства порожденных последовательностей, так и оптимизацию скорости вычисления с учетом архитектуры процессора. Читателю предлагается проверить какой-либо из рекомендуемых датчиков по схеме, представленной выше.

9.4.2 Датчик на основе линейной последовательнстной машины (ЛПМ)

Обозначим через $GF(2)$ поле вычетов по модулю 2. Выбираются натуральные числа n, m, k. Согласно [54], ЛПМ определяется матрицами

A, B, C, с элементами из поля $GF(2)$, где A $n \times n$ матрица, матрицы B, C имеют размерности $n \times m$ и $k \times n$ соответственно. Состояния ЛПМ суть векторы длины n $\mathbf{q} = \langle q_1, \ldots, q_n \rangle^T$, где $q_i \in GF(2)$, а входной и выходной алфавиты состоят из аналогичных векторов длины m и k соответственно. Состояние меняется согласно формуле

$$\mathbf{q_k} = A\mathbf{q_{k-1}} \oplus B\mathbf{x_k}, \tag{9.4.6}$$

а выходной сигнал вычисляется согласно

$$\mathbf{y_k} = C\mathbf{q_k} \tag{9.4.7}$$

В формулах (9.4.6),(9.4.7) все матричные операции осуществляются в поле $GF(2)$. Если требуется, чтобы на выходе генератора было число из интервала [0,1], выходной двоичный вектор $\mathbf{y}$ интерпретируют как двоичную запись числа $0.y_1, \ldots, y_k$. В случае автономной ЛПМ матрица $B = \Theta$, а состояние меняется согласно

$$\mathbf{q_k} = A\mathbf{q_{k-1}} \tag{9.4.8}$$

Очевидно, что если начальное состояние генератора (9.4.8) выбрано нулевым, то генератор не выйдет из этого состояния. Оказывается, что можно выбрать матрицу A таким образом, чтобы диаграмма переходов состояла лишь из двух компонент: нулевого вектора и кольцевой диаграммы переходов, куда входят все остальные состояния. В этом случае говорят, что генератор имеет максимальный период.

В [54] указано как осуществить выбор матрицы A, чтобы генератор имел максимальный период. Достоинством указанной схемы является ее простая аппаратная реализация. Действительно, предположим, что матрица

$$A = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & 0 & 1 \\ a_n & a_{n-1} & a_{n-2} & \ldots & a_1 \end{pmatrix} \tag{9.4.9}$$

Если $\mathbf{q_{k-1}} = \langle q_1, q_2, \ldots, q_n \rangle^T$, то согласно (9.4.8) $\mathbf{q_k} = \langle q_2, q_3, \ldots, f \rangle^T$, где

$$f = \oplus \sum_{i=1}^{n} a_i q_{n+1-i} \tag{9.4.10}$$

Изменения состояний ЛПМ указанного вид иногда удобнее рассматривать как последовательность, порождаемую рекуррентным соотношени-

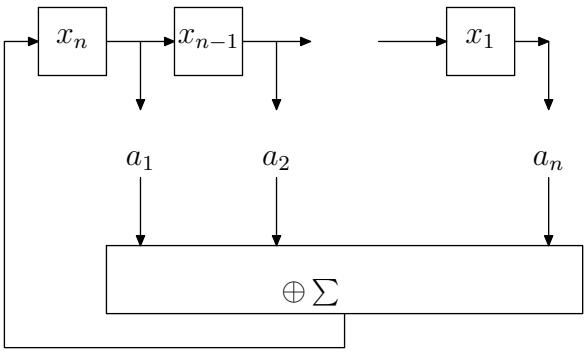


Рис. 9.2: Пример реализации датчика на основе регистра сдвига

ем. Имеются элементы $q_0, q_1, \ldots, q_{n-1} \in GF(2)$. Следующий элемент последовательности получается согласно формуле

$$q_k = \oplus \sum_{i=1}^{n} a_i q_{k-i},\ k = n, n+1, \ldots \tag{9.4.11}$$

Рассмотрим схему, представленную на Рис.9.2. Она состоит из n элементов памяти, организованных в виде регистра сдвига, умножителей на коэффициенты и комбинационной схемы $\oplus\sum$, суммирующей поступающие сигналы. Состоянием автомата является набор битов, находящихся в элементах памяти. В следующий момент времени происходит сдвиг содержимого регистров вправо, а элемент памяти x_n вычисляется согласно (9.4.10). При подходящем выборе матрицы A выполнены тесты на равномерность и на независимость в терминах автокорреляционной функции [54]. Различные реализации автоматов указанных типов и свойства порождаемых последовательностей представлены в [77] .

С математической точки зрения, не имеет значения, какая матрица будет обеспечивать максимальный период генератора. Однако, с точки зрения реализации полезно выбрать матрицу A таким образом, чтобы число ненулевых элементов в ней было минимальным. Действительно, каждый ненулевой элемент в этой матрице означает физическое соединение блоков. Нижнюю строку матрицы (9.4.9) принято задавать в виде многочлена

$$Pol(x) = x^n \oplus \sum_{k=1}^{n} x^n$$

Эти многочлены называются многочленами максимального показателя. В [54] приведены многочлены для некоторых n, для которых многочлен

$Pol(x)$ имеет только 3 ненулевых коэффициента. Например,

$$1 \oplus x \oplus x^3,\ 1 \oplus x^2 \oplus x^5,\ 1 \oplus x^5 \oplus x^{23}.\ 1 \oplus x^{11} \oplus x^{36}$$

Читателю предлагается оценить качество генератора, используя приведенную выше технику.

9.5 Применение генераторов псевдослучайных чисел для порождения криптографических ключей

Как было указано выше, последовательность порождаемая генератором псевдослучайных чисел, полностью определяется начальным состоянием. Для обеспечения "случайности"всей последовательности случайным образом порождают начальное состояние. В частности, в качестве начального состояния часто берут младшие разряды таймера. Однако, для целей криптографии важно обеспечить невозможность восстановления всей последовательности по ее части. Во всех рассмотренных выше случаях, вся последовательность легко восстанавливается по одному состоянию. Дело не спасает, если в качестве выхода генератора брать не все состояние, а только его часть, например, только один бит или линейную комбинацию битов над полем $GF(2)$. Пусть выход генератора (9.4.8) в момент времени k определяется формулой

$$y[k] = C\mathbf{g_k},$$

где $\mathbf{q_k} = \langle q_1, q_2, \ldots, q_n\rangle^T$-состояние генератора в момент времени k, а $C = (c_1, c_2, \ldots, c_n)$, $C \neq \Theta$. Заметим, что при анализе криптостойкости всегда считается известной структура генератора. В данном случае это означает, что известны матрицы A и C. Положим

$$\bar{A} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \ldots \\ CA^{n-1} \end{pmatrix}$$

Предположим, что имеет место равенство

$$\mathrm{rank}(\bar{A}) = n \tag{9.5.12}$$

Это условие называется условием наблюдаемости ЛПМ [54]. Если $\mathbf{q_0}$ начальное состояние, то столбец $\mathbf{y}$, составленный из первых n битов имеет

вид

$$\bar{A}\mathbf{q_0} = \mathbf{y}$$

В силу невырожденности матрицы $\bar{A}$ начальное состояние восстанавливается по первым n последовательным битам.

От указанного недостатка пытаются избавиться включением какой-либо нелинейности в структуру генератора. Простейшая схема, обеспечивающая цикличность, предложена Голомбом [19]. В терминах рекуррентных последовательностей (9.4.11) формула имеет вид

$$q_k = q_{k-n+1} \oplus f(q_{k-1}, \ldots, q_{k-n+2}),$$

где $f()$ произвольная булевская функция. Данная формула гарантирует цикличность каждой компоненты в диаграмме переходов. В этом случае не очевиден алгоритм восстановления начального состояния по выходной последовательности. В то же время возникают проблемы с доказательством статистических свойств выходной последовательности. Некоторые примеры исследования подобных регистров представлены в [67]. Обобщение формул Голомба, также обеспечивающих цикличность всех компонент, представлено в [80].

9.6 Комбинированные генераторы псевдослучайных чисел КГПСЧ

Несмотря на простоту генерации псевдослучайных криптографических ключей, в тех случаях, когда требуется высокая степень защиты, используют случайные ключи, порожденные некоторым физическим процессом. Как отмечалось выше, требуется, чтобы отдельные символы имели равномерное распределение и были независимыми. В данной главе рассмотрим ситуацию, когда некоторый физический датчик выдает независимые сигналы, но вероятность каждого из них колеблется в некоторых пределах.

9.6.1 Постановка задачи

Пусть имеется конечный автомат $\Xi(X, Y, Q, \delta, \lambda, q_0)$, где X, Y входной и выходной алфавиты соответственно, функция переходов $\delta : X \times Q \to Q$, функция выходов $\lambda : Q \to Y$, начальное состояние q_0. Имеется случайный процесс $\xi(t)$, принимающий значения из множества X, случайные величины $\xi(t_1), \xi(t_2)$ являются независимыми для $t_1 \neq t_2$, но имеют,

вообще говоря, разные распределения. Предполагается, что вероятность $p_x(t) = P(\xi(t) = x \in X)$ заключена в некотором интервале

$$0 < a_x \leq p_x \leq b_x < 1, \tag{9.6.13}$$

не зависящем от t. Случайные сигналы поступают на вход автомата Ξ, в результате последовательность состояний автомата $q(t)$ образует неоднородную цепь Маркова [44]. Идея выравнивания вероятностей с помощью автомата заключается в следующем [77]: выбирается натуральное m, и сигналы с выхода автомата снимаются с этим шагом. Полученные сигналы становятся случайными величинами. Оказалось, что при некоторых естественных предположениях вероятность появления любого символа алфавита Y на выходе становится одной и той же, а сами случайные величины, будут независимыми. Последние утверждения выполняются лишь с некоторой точностью. Строгое определение этого понятия будет дано ниже.

9.6.2 Случай линейной последовательностной машины

Рассмотрим ситуацию, когда в качестве автомата выбрана ЛПМ. Случай, когда в качестве ЛПМ был выбран регистр сдвига с обратными связями, подробно рассмотрен в [77]. Имеется ЛПМ, состояния которой есть векторы длины n над $GF(2)$. Для простоты изложения предположим, что входной алфавит Y состоит лишь из двух символов [0,1] и кодируется вектором длины 1. В силу этого матрица B в (9.4.6) будет столбцом длины n.. Пусть в момент времени t_s ЛПМ находится в состоянии $\mathbf{q_s}$. В этот момент с некоторой вероятностью p_0 на вход ЛПМ поступит сигнал 0 и с вероятностью p_1 поступит сигнал 1. Это означает, что с вероятностью p_0 $\mathbf{q_{s+1}} = A\mathbf{q_s}$ и с вероятностью p_1 $\mathbf{q_{s+1}} = A\mathbf{q_s} \oplus B$. С помощью этих соотношений строится матрица переходов $T(t_s)$ цепи Маркова . Это матрица размера $2^n \times 2^n$. Каждому состоянию ЛПМ отвечает одна строка и один столбец этой матрицы. Обычно состоянию $\langle q_1, \ldots, q_n \rangle^T$ ставится в соответственно номер строки или столбца, двоичный номер которых есть число $q_1 q_2 \ldots q_n$. Отметим, что номера строк и столбцов начинаются с 0. Обозначим через k, l, m номера строк, отвечающие состояниям $\mathbf{q_s}$, $A\mathbf{q_s}$, $A\mathbf{q_s} \oplus B$ соответственно. Матрица T имеет следующий содержательный смысл — из состояния с номером k ЛПМ перейдет в состояние с номером l с вероятностью p_0 и в состояние с номером m с вероятностью p_1, поэтому $T(t_s)[k|l] = p_0$, $T(t_s)[k|m] = p_1$. Чтобы сделать рассуждения более

понятными, рассмотрим пример. Пусть $n = 3$,

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Согласно определению

$$T = \begin{pmatrix} p_0 & p_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_0 & p_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_1 & p_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_1 & p_0 \\ p_1 & p_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p_1 & p_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p_0 & p_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_0 & p_1 \end{pmatrix}$$

Поясним вид строки с номером 1 матрицы T(напомним, что нумерация начинается с 0). Эта строка соответствует состоянию $\mathbf{q_1} = \langle 0, 0, 1\rangle^T$. Находим $A\mathbf{q_1} = \langle 0, 1, 0\rangle^T = \mathbf{q_2}$, и $A\mathbf{q_1} \oplus B = \mathbf{q_3}$. Это объясняет появление чисел p_0 и p_1 в столбцах с номерами 2 и 3.

Как и следовало ожидать, сумма элементов в каждой строке матрицы T равна 1, то есть матрица является стохастической. Это имеет место для любой матрицы переходов цепи Маркова. Однако, в данном случае и сумма элементов в каждом столбце матрицы T также равна 1, то есть матрица является дважды стохастической. Последнее свойство справедливо для любой ЛПМ с невырожденной матрицей A. Действительно, в рассматриваемом случае выберем произвольное состояние $\bar{\mathbf{q}}$. Тогда для любого входного сигнала $\mathbf{x}$, равного значению случайного процесса $\xi(t)$, всегда существует решение $\mathbf{q}$ уравнения

$$A\mathbf{q} \oplus B\mathbf{x} = \bar{\mathbf{q}}$$

Это означает, что сумма элементов в столбце, отвечающем состоянию $\bar{\mathbf{q}}$, равна сумме вероятностей значений случайной величины $\xi(t)$, то есть равна 1.

Пусть известна реализация процесса ξ длины n, то есть известны случайные величины $\xi(0), \xi(1), \ldots, \xi(n-1)$. В результате ЛПМ перейдет из состояния $\mathbf{q_0}$ в состояние

$$\mathbf{q_n} = A^n\mathbf{q_0} \oplus A^{n-1}B\xi(0) \oplus A^{n-2}B\xi(1) \oplus \cdots \oplus B\xi(n-1) \qquad (9.6.14)$$

Введем дополнительное ограничение на матрицы A и B. Предположим, что выполнено равенство

$$rank(A^{n-1}B, A^{n-2}B, \ldots, B) = n \tag{9.6.15}$$

Это свойство называют управляемостью системы [54]. Данное условие равносильно следующему – векторы вида

$$A^{n-1}B, A^{n-2}B, \ldots, B$$

составляют базу в пространстве столбцов длины n. Отсюда и из формулы (9.6.14) следует, что вектор $\mathbf{q_n}$ с положительной вероятностью может совпадать с любым столбцом длины n. С другой стороны, из определения цепи Маркова вытекает, что матрица

$$F_n = T(0)T(1)\cdots T(n-1)$$

дает распределение состояний автомата через n шагов. Строка $\langle 0, \ldots 0, 1, 0 \ldots 0\rangle F_n$, где 1 стоит в позиции с номером k, состоит из распределения вероятностей состояний через n шагов, если первоначально систем находилась в состоянии с номером k. Как было указано выше, при сделанных предположениях ЛПМ может перейти за n шагов из любого состояния в любое состояние с положительной вероятностью. Другими словами, все элементы матрицы F_n положительны. Легко проверяется, что произведение двух дважды стохастических матриц есть снова дважды стохастическая матрица. Теперь из положительности элементов матрицы F_n вытекает, что

$$F_n^m \to T_\infty,\ m \to \infty, \tag{9.6.16}$$

где T_∞ – матрица ранга 1 [44]. Поскольку это дважды стохастическая матрица, все ее элементы равны $1/2^n$. Выберем m настолько большим, чтобы разность между элементами матриц T_∞ и F_n^m была меньше заданного числа Δ. Как было указано выше, матрицы $T(k)$ могут меняться вместе с k, поэтому величина m также будет варьироваться вместе с реализацией. Однако, в силу (9.6.13) можно получить оценку, не зависящую от реализации [70]. Положим $s = nm$. Если съем сигнала с выхода ЛПМ производится с шагом s, то можно утверждать, что с указанной точностью вероятность попадания ЛПМ в любое состояние будет одной и той же и не будет зависеть от предыдущего состояния, с которого снимался сигнал.

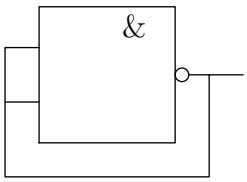


Рис. 9.3: Пример генератора на основе инвертора

9.7 Марковская модель генератора случайных чисел

9.7.1 Модель физического генератора случайных чисел

До сих пор физический источник случайных чисел оставался в стороне. В данном пункте будет описан один из таких источников, основанный на работе комбинационной схемы в нештатном режиме. Вероятно впервые, идея создания таких генераторов была предложена в [76] для комбинационных схем специального вида — линейных схем. Оказалось, что теория таких генераторов справедлива для схем весьма общего вида [81]. Эта теория представлена в данном пункте. Особенность теории генераторов на основе линейных схем будет изложена в следующем пункте.

Рассмотрим простейшую схему, изображенную на Рис.9.3. Она содержит инвертор на основе элемента И-НЕ, выход которого подключен к входу. С точки зрения обычной логики выход устройства не определен. Если же речь идет о физическом устройстве, то тут ситуация иная. Если на вход инвертора поступает единичный сигнал, то этот сигнал преобразуется в нулевой сигнал на выходе, но это преобразование происходит не мгновенно, а с некоторой задержкой. Аналогичный эффект имеет место, если на входы схемы подан нулевой сигнал. Это явление получило название "дребезжание". Предлагается математическая модель этого явления, которая заключается в том, что время t изменения выходного сигнала после изменения входного сигнала подчиняется экспоненциальному закону, то есть $P(t < t_0) = 1 - exp(-at_0)$, где a – параметр распределения. Это распределение обладает следующим замечательным свойством: $P(t < t_1 | t > t_0) = 1 - exp(-a(t_1 - t_0))$.

Содержательно последнее равенство означает следующее: если событие не произошло до момента времени t_0, то после этого момента время ожидания события подчиняется тому же самому закону. Это свойство марковости процесса, поскольку дальнейшее его поведения после момента времени t_0 зависит только от состояния процесса в этот момент времени и не зависит от предыдущей истории. На этом свойстве основаны

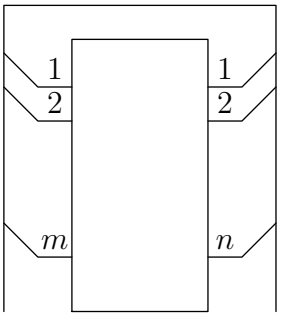


Рис. 9.4: Произвольная комбинационная схема в нештатном режиме

уравнения Эрланга, с помощью которых в дальнейшем будет описано поведение генератора.

9.7.2 Уравнения Эрланга для описания поведения "дребезжащей"схемы

Рассмотрим схему, изображенную на Рис.9.4. Имеется комбинационная схема с n выходами и m входами, причем выходы произвольным образом соединены с входами этой схемы (допускается, что на некоторые входы подан постоянный сигнал, а на несколько входов подан сигнал с одного выхода). В результате возможно, что схема перейдет в некоторое стабильное состояние (например, известная схема RS триггера), но возможно, что схема окажется в режим генерации. Будем рассматривать только второй случай. Сделаем следующие предположения [81]:

1. Время изменения сигнала на каждом выходе в результате изменения сигнала на входе, если это определяется логикой схемы, подчиняется экспоненциальному закону с одним и тем же параметром для всех выходов

2. В любой момент времени может измениться только один выход

3. Изменения сигналов на выходах являются независимыми событиями

Сделанные предположения о функционировании схемы позволяют описать динамику работы генератора с помощью дифференциальных уравнений, аналогичных уравнениям Эрланга в теории массового обслуживания (см., например, [83]). Составление этих уравнения проиллюстрируем на примере схемы, изображенной на Рис. 9.4, когда $m = n = 3$. Назовем состоянием $\mathbf{S(t)}$ генератора в момент времени t вектор $< y_1, y_2, y_3 >$, составленный из выходов трех элементов. На выходе с номером i реализуется булевская функция $f_i(y_1, y_2, y_3)$. Состояние гене-

ратора в следующий момент времени может измениться в результате изменения одного из выходов. Все возможные двоичные векторы-состояния перенумеруем, используя естественную двоичную нумерацию. Обозначим через $P_n(t)$ вероятность того, что в момент времени t генератор находится в состоянии с номером n. Рассмотрим некоторое состояние $< y_1, y_2, y_3 >$. Согласно сделанным предположениям, из данного состояния возможен переход в состояния $< f_1(y_1, y_2, y_3), y_2, y_3 >$, $< y_1, f_2(y_1, y_2, y_3), y_3 >$, и $< y_1, y_2, f_3(y_1, y_2, y_3) >$. Составим матрицу A переходов генератора. Как обычно, элемент этой матрицы, стоящий в строке с номером i и столбце с номером j, обозначим через $A[i|j]$. По определению этот элемент равен количеству переходов из состояния с номером i в состояние с номером j в результате изменения сигналов на выходах. При составлении этой матрицы учитываются также "виртуальные переходы". Под виртуальным переходом мы понимаем ситуацию, когда при вычислении текущего значения булевской функции на выходе значение функции не меняется. Например, если $y_1 = f_1(y_1, y_2, y_3)$, то считается, что в результате такого виртуального перехода состояние генератора не изменилось. Виртуальным переходам отвечают диагональные элементы матрицы A. В силу сделанного замечания, сумма элементов в каждой строке матрицы A равна числу выходов схемы, поскольку учитываются как обычные переходы, так и виртуальные переходы.

В рассматриваемом случае схема имеет 3 выхода, и генератор может находиться в одном из 8 состояний. Вероятность того, что в момент времени $t + \Delta t$ генератор будет в состоянии с номером n величина $P_n(t + \Delta t)$ складывается из из вероятности того, что в момент времени t система находилась в том же состоянии, и после этого не сработал ни один выход, и из вероятностей нахождения в момент времени t в другом состоянии и срабатывании подходящего выхода. Учтем предположение об экспоненциальном распределении времени всех срабатываний. Отбрасывая величины порядка больше первого по Δt, получим, что вероятность не срабатывания всех выходов равна $1 - 3a\Delta t$, а вероятность срабатывания одного выхода равна $a\Delta t$. Имеем

$$P_n(t + \Delta t) = P_n(t)(1 - 3a\Delta t) + \sum_{k=0}^{7} a\Delta t A[k|n] P_k(t)$$

Деля на Δt и переходя к пределу при $\Delta t \to 0$, получим

$$\frac{dP_n(t)}{dt} = -3aP_n(t) + a\sum_{k=0}^{7} A[k|n]P_k(t), \quad n = 0, ..., 7$$

В общем случае, когда генератор имеет m выходов и $r = 2^m$ состояний, уравнение принимает вид

$$\frac{dP_n(t)}{dt} = -maP_n(t) + a\sum_{k=0}^{r-1} A[k|n]P_k(t), \quad n = 0, ..., r-1 \qquad (9.7.17)$$

Заметим, что в (9.7.17) все виртуальные срабатывания исключаются. Если, например, при $m = 3$ имеются два виртуальных срабатывания в состоянии с номером n, то вероятность не срабатывания будет равна $1 - a\Delta t$. С учетом виртуальных срабатываний вероятность не срабатывания равна $1 - 3a\Delta t$, а $A[n|n] = 2$, и в результате, получается то же самое уравнение (9.7.17).

9.7.3 Финальный вектор марковского процесса

Обозначим через $\mathbf{P}(t) = \langle P_0(t), \ldots, P_{r-1}(t)\rangle$. Уравнение (9.7.17) в матричной форме имеет вид

$$\frac{d\mathbf{P}(t)}{dt} = a\mathbf{P}(t)(A - mI),$$

а его решение равно [44]

$$\mathbf{P}(t) = \mathbf{P}(0)e^{a(A-mI)t} \qquad (9.7.18)$$

По построению

$$\sum_{i=0}^{N-1} A[k|i] = m, \quad k = 0, \ldots, r-1$$

и все элементы этой матрицы неотрицательны. Отсюда следует, что A/m – стохастическая матрица, m будет характеристическим числом матрицы A. Как было указано выше, все остальные характеристические числа b_i матрицы A удовлетворяют неравенствам

$$|b_i| \le m, \quad i = 1, \ldots, r$$

В дальнейшем будем предполагать, что $b_1 = m$. Потребуем теперь, чтобы матрица A была неразложимой. Как следует из Теоремы 1, в этом случае число m будет простым корнем, поэтому справедливы неравенства

$$\mathrm{Re}(b_i) < m, \quad i = 2, \ldots, r, \qquad (9.7.19)$$

а числу m отвечает собственный вектор-строка $\mathbf{Q} =< q_1, \ldots, q_N >$ с положительными элементами: $m\mathbf{Q} = \mathbf{Q}A$. Без ограничения общности можем предполагать, что вектор $\mathbf{Q}$ нормирован условием $\sum_k q_k = 1$. В матрице $A - mI$ одно характеристическое число равно нулю, а остальные характеристические числа $b_i - m$ имеют, согласно (9.7.19), отрицательные вещественные части. Положим $z_i = \exp(b_i - m)$. Имеем

$$|z_i| < 1, \quad i = 2, \ldots, N \tag{9.7.20}$$

Из теории функций от матриц (см. [44]) известно, что числа z_i будут характеристическими числами матрицы $\exp(A - mI)$. Жорданова форма матрицы $\exp((A - mI)t)$ имеет вид

$$\exp((A - mI)t) = T^{-1} diag(J_0, J_1, \ldots, J_L)T \tag{9.7.21}$$

Здесь T – матрица, не зависящая от t, $J_0 = (1)$ – жорданова клетка первого порядка, а $J_i, \quad i > 0$ – жорданова клетка, построенная по некоторому $z_j^t, \quad j > 1$. Из (9.7.20) теперь вытекает, что матрица $\exp(t(A - mI)a)$ стремится к матрице ранга 1, когда $t \to \infty$, поскольку все клетки $J_i, \quad i > 0$ стремятся к нулевым матрицам. Вектор $\mathbf{Q}$ останется левым собственным вектором и для матрицы $exp((A - mI)t)$ и отвечает собственному значению 1, поэтому он будет собственным вектором и для предельной матрицы. С другой стороны, матрица $exp((A - mI)t)$ будет стохастической, поэтому и предельная матрица будет тоже стохастической. Это означает, что все строки предельной матрицы совпадают с вектором $\mathbf{Q}$. Теперь из (9.7.18) следует, что

$$\mathbf{P}(t) \to \mathbf{Q}, \quad t \to \infty$$

то есть предельное распределение не зависит от начального распределения вероятностей $\mathbf{P}(\mathbf{0})$.[44]. Содержательно вектор $\mathbf{Q}$ представляет вероятности нахождения системы в каждом состоянии, когда время стремится к бесконечности, а сам вектор называется финальным вектором для соответствующего марковского процесса. Обозначим через F_Q матрицу, все строки которой совпадают с вектором $\mathbf{Q}$. По любому числу ϵ можно найти такое число q, что

$$||\exp(a(A - mI)t) - F_Q|| < \epsilon, \ t > q$$

Определяя состояния генератора с шагом q, получаем распределение вероятностей состояний близкое к финальному с заданной точностью. Это позволяет дать количественную оценку "независимости" полученного отсчета от предыдущего. Зная явный вид матрицы A, можно получить и оценку для q. Ниже будет приведен пример получения такой оценки.

9.7.4 Примеры вычисления финальных векторов для нелинейных схем

Все вычисления в данном пункте проведены с помощью пакета математических программ SciLab [36]. Вернемся к схеме, изображенной на Рис.9.4. Определим следующим образом булевы функции, реализуемые схемой. Пусть $i_0, \ldots, i_7$ - произвольное множество из чисел $i_k \in [0, 7]$. Каждое число $k \in [0, 7]$ в двоичной форме представимо тремя битами. Когда эти биты поданы на вход схемы, на выходе появляются биты, отвечающие числу i_k. Например, если $i_2 = 4$, то при подаче на вход схемы вектора $\langle 0, 1, 0\rangle$, на выходах появится вектор $\langle 1, 0, 0\rangle$. Очевидно, что любая комбинационная схема с тремя входами и тремя выходами может быть задана последовательностью $i_0, \ldots, i_7$. Предполагается, что выполнены все условия, наложенные на схему, указанные выше. В первом примере последовательность имеет вид $1, 2, 3, 4, 5, 6, 7, 0$. В этом случае матрица

$$A_1 = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Непосредственной проверкой можем убедиться, что элементы матрицы $(I + A_1)^5$ положительны, следовательно, это неразложимая матрица. В силу симметричности матрицы A_1, вектор $\mathbf{Q} = \langle 1, 1, 1, 1, 1, 1, 1, 1\rangle/8$. Это означает, что финальная вероятность каждого состояния будет одной и той же. В следующем примере последовательность имеет вид $1, 3, 5, 7, 6, 4, 2, 0$. Имеем

$$A_2 = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Для отыскания финального вектора в SciLab сначала вводим матрицу A_2, после чего выполняем команду

```
[a,b]=spec(A2')
```

Сначала будет напечатана диагональная матрица b, на диагонали которой находятся характеристические числа матрицы. Столбцы матрицы a есть собственные векторы матрицы A_2^T (знак $'$ означает транспонирование матрицы в SciLab), то есть получаем собственные векторы-строки исходной матрицы. Тот вектор, который будет отвечать собственному значению m и будет финальным вектором. Он равен $\langle 0.0833, 0.0833, 0.0833, 0.2500, 0.0833, 0$
Практическое значение имеют не вероятности отдельных состояний, а вероятности появления 0 или 1 на отдельном выходе. Эти вероятности могут быть найдены по вероятностям появлений состояний из финального вектора путем суммирования вероятностей состояний, имеющих 0 или 1 в одной позиции. Например, найдем вероятность появления 0 на первом выходе. Для этого надо суммировать финальные вероятности состояний $\langle 0,0,0\rangle, \langle 0,0,1\rangle, \langle 0,1,0\rangle, \langle 0,1,1\rangle$, то есть вероятности состояний с номерами 0,1,2,3. Нетрудно видеть, что эта сумма равна 0.5. Аналогичный результат получается и для третьего выхода. С другой стороны, для подсчета вероятности появления 0 на втором выходе, надо суммировать финальные вероятности состояний с номерами 0,1,4,5. Теперь эта сумм равна $\frac{1}{3}$. Еще одни пример, когда последовательность выбрана в виде $2,4,6,7,5,3,1,0$. Финальный вектор в этом случае имеет вид $\langle 0.0811, 0.0811, 0.1892, 0.1351, 0.1081, 0.1622, 0.1081, 0.1351\rangle$. Здесь вероятность 0 на первом выходе равна 0.4865, вероятность 0 на втором выходе равна 0.4324, вероятность 0 на третьем выходе равна 0.4865.

Следующий пример относится к схеме, представленной на Рис.9.7.4. Здесь изображена кольцевая схема, составленная из двух схем, рассмот-

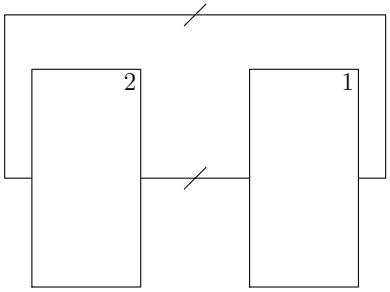


Рис. 9.5: Кольцевое соединение двух схем

ренных выше. Для первой схемы последовательность равна $1,2,3,4,4,3,2,1$, а для второй схемы она равна $1,3,5,7,6,4,2,0$. В этом случае схема имеет 64 состояния. Каждое состояние есть двоичный вектор длины 6, в котором первые 3 компонента - выходы первой схемы, а оставшиеся 3 компо-

нента - выходы второй схемы. Выбор множеств обусловлен необходимостью обеспечить неразложимость матрицы A. Приведем вероятности 0 на выходах первой схемы. Это 0.7652, 0.5369, 0.4472; вероятность появления 0 на выходе второй схемы равны 0.4159, 0.4117, 0.2348. Применение генераторов кольцевой структуры позволяет использовать много вариантов соединений схем друг с другом, не меняя самих схем. Это свойство может оказаться полезным при создании перестраиваемых генераторов криптографических ключей Одной из нерешенных проблем остается метод выбора компонентов и их соединений для обеспечения неразложимости матрицы переходов A. Каждый раз приходится проверять ее, исходя из определения.

9.7.5 Оценка близости текущих вероятностей к финальным

Матрица A по текущей схеме строится достаточно просто. Ограничимся случаем, когда в пространстве существует базис из собственных векторов матрицы A. Случай произвольной жордановой формы рассматривается аналогично, хотя требует более громоздких вычислений. Зная вид этой матрицы можно получить оценку нормы разности

$$|| \exp(a(A - mI)t) - F_Q||$$

На примере матрицы A_2 покажем, как это можно сделать. Без ограничения общности можно считать $a = 1$. Напомним, что характеристические числа обозначались символами $b_k, b_1 = 3$. Прямым вычислением находим, что $\max_{k>1} Re(b_k) = 2.4196$. Собственный вектор $\mathbf{Q}$, отвечающий собственному значению 3, был найден выше. Вычитая из всех характеристических чисел 3, получим, что в матрице $A_2 - 3I$ одно характеристическое число нулевое, а наибольшая вещественная часть остальных чисел равна -0.5804. Следовательно, среди характеристических чисел матрицы $\exp(A_2 - 3I)$ одно число равно 1, а наибольший модуль оставшихся чисел равен $d = \exp(-0.5804) = 0.55597$. В данном примере матрица A_2 имеет простой спектр, поэтому в пространстве существует базис из собственных векторов этой матрицы, а соотношение (9.7.21) принимает вид

$$A_2 = T^{-1} diag(b_1, b_2, \ldots, b_8)T,$$

где T – матрица, столбцы которой есть собственные векторы матрицы A_2. Последнее равенство перепишем в следующей форме

$$A_2 = T^{-1} diag(b_1, 0, \ldots, 0)T + T^{-1} diag(0, b_2, \ldots, 0)T + \cdots + T^{-1} diag(0, 0, \ldots, b_8)T.$$

Вводя обозначение $B_k = T^{-1}diag(0,\ldots,b_k,0,\ldots,0)T$, перепишем предыдущее выражение в виде

$$A_2 = \sum_{k=1}^{8} b_k B_k$$

Это равенство известно как спектральное разложение матрицы [53]. Теперь

$$\exp(A_2 - 3I) = F_Q + \sum_{k=2}^{8} z_k B_k$$

где матрица F_Q была определена выше. Поскольку после возведения матрицы в целую степень q ее характеристические числа возводятся в ту же степень, а собственные векторы при этом не меняются, получаем

$$\exp((A_2 - 3I)r) = F_Q + \sum_{k=2}^{8} z_k^r B_k \tag{9.7.22}$$

или

$$||\exp((A_2 - 3I)r) - F_Q|| \leq \sum_{k=2}^{8} |z_k|^r ||B_k||$$

Фактически, основным параметром, определяющим скорость установления, является число d, найденное выше. Чем меньше значение этого параметра, тем быстрее происходит процесс установления финального распределения.

9.8 Генератор случайных чисел на основе сумматоров по модулю два

9.8.1 Основные определения

В предыдущем пункте был изложен общий подход к теории генераторов на основе комбинационных схем, работающих в режиме "дребезжания". В данном параграфе будет рассмотрен частный случай, когда комбинационная схема состоит из сумматоров по модулю два. Исторически именно эти схемы были впервые предложены и рассмотрены в связи с созданием генераторов случайных чисел в работе [76]. Подробное изложение вопросов, связанных со схемной реализацией таких устройств, представлено в [77]. В данном пункте остановимся на особенностях теории функционирования подобных генераторов [63]. Рассмотрим схему, представленную на Рис. 9.6 На свободный вход сумматора с номером 3 подается постоян-

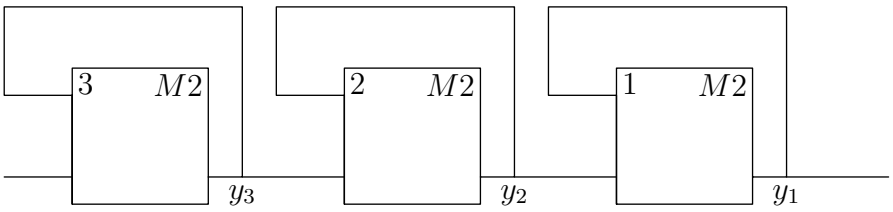


Рис. 9.6: Пример генератора, составленного из сумматоров по модулю два

ный сигнал 1. Согласно подходу, изложенному выше, состояние схемы в момент времени t задается вектором $\mathbf{S(t)} = \langle y_1, y_2, y_3 \rangle^T$. Очевидно, что состояние генератора будет все время меняться.

9.8.2 Математическая модель

Уточним математическую модель устройства, рассмотренную ранее, в случае схемы, составленной из сумматоров по модулю 2. В дальнейшем будет говорить просто о сумматорах. В общем случае схема состоит из произвольного количества сумматоров, выходы которых соединены с входами других сумматоров или тех же самых сумматоров; сигнал с выхода одного сумматора может подаваться на входы нескольких устройств; на часть входов могут подаваться постоянные сигналы. Теперь предположения, относящиеся к общим принципам функционирования подобных устройств, выглядят следующим образом

1. Время срабатывания каждого сумматора меняется согласно экспоненциальному закону с одним и тем же параметром
2. В любой момент времени может сработать только один сумматор
3. Срабатывания сумматоров являются независимыми случайными величинами

Система уравнений типа Эрланга, описывающая динамику генератора, остается той же самой. В этом плане никаких новых эффектов, связанных с ограничениями на вид изучаемых комбинационных схем, не наблюдается. То же самое относится и к оценке времени установления генератора. Все особенности проявляются, когда переходят к вопросу о существовании стационарных состояний. Рассмотрим схему, представленную на Рис.9.7. На свободный вход схемы с номером 2 подан единичный сигнал. Нетрудно видеть, что состояние $\langle 1, 0 \rangle^T$ будет стабильным для устройства — из этого состояния устройство без внешнего воздействия выйти не может. Очевидно, существование стабильных состояний в схеме является

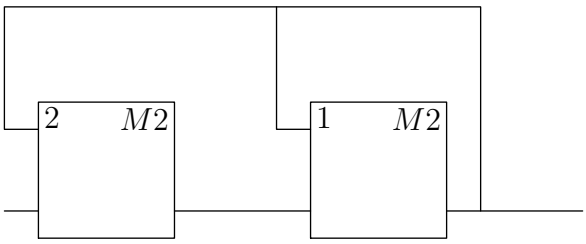


Рис. 9.7: Пример схемы со стабильными состояниями

нежелательным моментом. Формально, существование таких состояний легко выявить по матрице A. Состояние с номером i будет стабильным тогда и только тогда, когда в строке матрицы с этим номером лишь элемент $A[i|i]$ отличен от нуля. Однако, если комбинационная схема имеет n выходов, матрица A имеет размер $2^n \times 2^n$ и становится мало пригодной для теоретического исследования. В случае схемы, составленной из сумматоров, существует более простой способ для выявления таких состояний.

9.8.3 Стабильные и частично стабильные состояния схемы, составленной из сумматоров

При изложении этого пункта будем следовать работе [64]. Кроме стабильных состояний, о которых шла речь выше, устройство может обладать частично стабильными состояниями. Если генератор попадает в такое состояние, то в дальнейшем выходы некоторых сумматоров остаются неизменными, хотя само состояние стабильным не будет. Очевидно, что существование таких состояний ухудшает качество генератора. Если наличие стабильных состояний легко выявляется по матрице A, то доказательство отсутствия частично стабильных состояний в схеме решается более сложно. Перенумеруем все сумматоры целыми числами от 1 до n. На свободные входы сумматоров поступают постоянные сигналы. Существование сумматора, на оба входа которого поступают постоянные сигналы, не имеет смысла. Если постоянный сигнал поступает на вход сумматора с номером k, то этот сигнал обозначим через b_k. Выход сумматора с номером k равен y_k, а состояние генератора в момент времени t есть вектор, состоящий из выходов всех сумматоров, $\mathbf{S(t)} = \langle y_1, \ldots, y_n \rangle^T$.

9.8.4 Альтернативный способ описания структуры генератора

Воспользуемся линейностью компонентов комбинационной схемы. Это дает альтернативный способ описания структуры генератора. Напомним, что все арифметические операции выполняются над полем $GF(2)$.

Сигнал y_i на выходе сумматора с номером i определяется значениями на входах, которые порождаются сумматорами, и некоторыми постоянными сигналами. В результате срабатывания сумматора с номером i может измениться лишь сигнал y_i. На входы этого сумматора могут поступать сигналы с сумматоров с номерами p и q либо сигнал с сумматора с номером p и постоянный сигнал b_i. Исключается возможность $p = q$, поскольку выход сумматора с номером i в этом случае окажется нулевым. Если сумматор с номером i не имеет свободных входов, полагаем $b_i = 0$. В результате срабатывания в первом случае значение y_i заменится на $y_p \oplus y_q \oplus b_i$. а во втором — на $y_p \oplus b_i$. Определим вектор $\mathbf{B} = \langle b_1, b_2, \ldots, b_n \rangle^T$ и матрицу L размера $n \times n$ следующим образом: в строке с номером i в первом случае $L[i|p] = L[i|q] = 1$, а остальные элементы в этой строке нулевые, а во втором случае только элемент $L[i|p] = 1$, а остальные элементы строки нулевые. Обратим внимание, что имеется дополнительное ограничение на элементы матрицы L и вектора $\mathbf{B}$, которое всегда считается выполненным — если в строке $L[i|*]$ два ненулевых элемента, то $b_i = 0$. Изменение состояния в результате срабатывания сумматора с номером i определяется равенством

$$y_i' = L[i|*]\mathbf{S} \oplus b_i \tag{9.8.23}$$

Другими словами, матрица L и вектор $\mathbf{B}$ определяют структуру схемы и ее функционирование. В качестве примера рассмотрим схему на рис. 9.6. Для этого генератора матрицы L и $\mathbf{B}$ имеют вид

$$L = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Поскольку дальнейшие результаты формулируются в терминах строк матрицы L, введем для этих строк специальные обозначения – $\lambda_i = L[i|*]$. В этих обозначениях

$$L = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \cdots \\ \lambda_n \end{pmatrix}$$

Матрицу L назовем структурной матрицей генератора.

9.8.5 Стабильные состояния

Легко видеть, что при нулевом векторе $\mathbf{B}$ нулевое состояние всегда является стабильным. Интерес представляет ситуация, когда этот вектор отличен от нулевого, что и будет предполагаться в дальнейшем.

Предложение 3 *Пусть*

$$D = L \oplus I = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \cdots \\ \lambda_n \end{pmatrix} \oplus I \quad , \bar{D} = \begin{pmatrix} D[1|*], & b_1 \\ D[2|*], & b_2 \\ \cdots & \cdots \\ D[n|*], & b_n \end{pmatrix},$$

где I – единичная матрица. Тогда генератор обладает стабильными состояниями тогда и только тогда, когда

$$\mathrm{rank}(D) = \mathrm{rank}(\bar{D}), \tag{9.8.24}$$

где $\mathrm{rank}(D)$ *означает ранг матрицы D.*

Доказательство. Согласно (9.8.23), состояние $\mathbf{S}$ будет стабильным тогда и только тогда, когда

$$y_i = \lambda_i \mathbf{S} \oplus b_i, \quad i = 1, \ldots, n$$

В матричной форме это условие перепишется в виде

$$\mathbf{S} = L\mathbf{S} \oplus \mathbf{B}. \tag{9.8.25}$$

Учитывая то, что все операции осуществляются в поле $GF(2)$, запишем равенство (9.8.25) в форме

$$(I \oplus L)\mathbf{S} = \mathbf{B}$$

Другими словами, вектор $\mathbf{S}$ является решением неоднородной системы. Для того, чтобы такая система была совместной, согласно теореме Кронекера-Капелли необходимо и достаточно выполнения условия (9.8.24).

9.8.6 Частично стабильные состояния

Согласно Предложению 3, выполнение неравенства $\mathrm{rank}(D) < \mathrm{rank}(\bar{D})$ гарантирует отсутствие стабильных состояний. В то же время возможна ситуация, когда состояние не является стабильным, но при этом остаются постоянными выходы некоторых сумматоров, то есть состояние является частично стабильным. Пусть в частично стабильном состоянии

$\mathbf{S} = \langle y_1, \ldots, y_n \rangle^T$ на выходах сумматоров с номерами $i_1, \ldots, i_k$, $0 < k < n$, сигналы не меняются в процессе работы генератора. Поскольку нумерация сумматоров является произвольной, можем считать, что в процессе работы генератора сигналы на выходах сумматоров с номерами i, $i = 1, \ldots, k$ не меняются. Частично стабильное состояние представим в виде $\mathbf{S} = \langle \mathbf{S_1}, \mathbf{S_2} \rangle^T$, где $\mathbf{S_1} = \langle y_1, \ldots, y_k \rangle$, $\mathbf{S_2} = \langle y_{k+1}, \ldots, y_n \rangle$.

Согласно сделанным предположениям, имеют место равенства

$$\lambda_i \mathbf{S} \oplus b_i = y_i, i = 1, \ldots, k \tag{9.8.26}$$

при любых изменениях в компонентах вектора $\mathbf{S_2}$. Положим $\epsilon_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, где 1 занимает позицию с номером i. Изменение компоненты с номером $j, k < j < n$, на противоположный не должно влиять на компоненту с номером $i, 0 < i \leq k$. Указанное изменение реализуется заменой вектора $\mathbf{S}$ на $\mathbf{S} \oplus \epsilon_j$. Из (9.8.26) вытекает, что

$$\lambda_i \epsilon_j = 0, \quad j = k+1, \ldots, n; i = 1, \ldots, k. \tag{9.8.27}$$

Условие (9.8.27) означает, что при указанной нумерации сумматоров

$$L = \begin{pmatrix} C_1 & 0 \\ C_2 & C_3 \end{pmatrix},$$

где C_1 есть $k \times k$ матрица. То есть, матрица оказывается разложимой.

Предложение 4 *Для того, чтобы генератор обладал частично стабильными состояниями необходимо выполнение равенств (9.8.26), из которых следует, что структурная матрица генератора является разложимой.*

Итак, условие разложимости структурной матрицы является необходимым условием для существования частично стабильных состояний генератора. Матрица L будет неразложимой тогда и только тогда, когда все элементы матрицы $(I+L)^{n-1}$ будут положительны (при вычислении степени матрицы все операции осуществляются в поле вещественных чисел). Это дает простой способ проверки неразложимости матрицы переходов.

Разложимость матрицы L обеспечивает отсутствие влияния срабатывания сумматоров с номерами $k+1, \ldots, n-1$ на выходы сумматоров с номерами $i = 1, \ldots, k$. Однако, это условие не является достаточным для частичной стабильности состояния, поскольку не учитывается влияние срабатывания сумматоров с номерами $i = 1, \ldots, k$ на данное состояние.

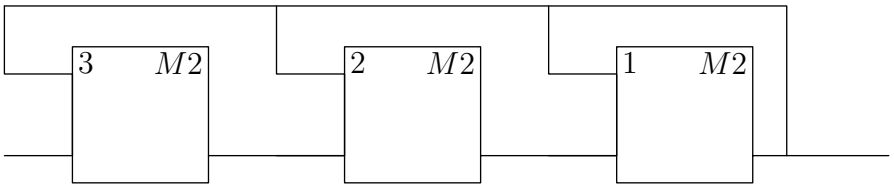


Рис. 9.8: Пример схемы с глобальной обратной связью

9.8.7 Примеры

В приведенных выше примерах каждый сумматор имел ровно два входа, поэтому структурная матрица генератора имела не более двух единиц в каждой строке. На самом деле, это ограничение является несущественным, можно рассматривать сумматоры с числом входов больше, чем два, поэтому число единиц в каждой строке матрицы переходов может быть произвольным.

В качестве иллюстрации рассмотрим ситуацию со стабильными состояниями для схемы на Рис. 9.8. На свободный вход сумматора 3 подан единичный сигнал. Для этой схемы

$$L = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \quad \bar{D} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Нетрудно видеть, что $\mathrm{rank}(D) = 2$ и $\mathrm{rank}(\bar{D}) = 3$ над полем $GF(2)$, и в схеме отсутствуют стабильные состояния. Над полем вещественных чисел

$$D^2 = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{pmatrix}.$$

Это означает, что структурная матрица является неразложимой, поэтому в схеме отсутствуют частично стабильные состояния. В данном случае справедливость полученных утверждений можно проверить непосредственно. Более сложный пример представляет кольцевая схема, состоящая из n сумматоров, когда на один вход сумматора с номером i поступает сигнал с сумматора с номером $i+1, i = 1, \ldots, n-1$, а на вход последнего сумматора поступает сигнал с первого сумматора. На второй вход каждого сумматора поступает либо постоянный сигнал, либо выход

любого сумматора. Легко проверить, что матрица перестановки

$$C = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

является неразложимой. Структурная матрица L генератора получается добавлением некоторого числа единиц в строки матрицы C, поэтому матрица L также будет неразложимой. Это означает, что такая схема не может иметь частично стабильных состояний. В частности, рассмотрим схему, в которой $L = C$, то есть на один из входов каждого сумматора подается постоянный сигнал. Матрица $D = L \oplus I$ имеет в каждой строке две единицы, поэтому сумма всех столбцов матрицы есть нулевой вектор. Это означает, что $\mathrm{rank(D)} < \mathrm{n}$. Выбрав произвольный вектор $\mathbf{B}$ с нечетным числом единиц, получим, что $\mathrm{rank(D)} < \mathrm{rank}(\bar{\mathrm{D}})$, поскольку такой вектор $\mathbf{B}$ нельзя получить в виде линейной комбинации столбцов, в каждом из которых четное число единиц. Полученная схему не будет иметь как стабильных, так и частично стабильных состояний.

Другой крайний случай – когда схема не имеет глобальной обратной связи (пример такой схемы для $n = 3$ представлен на рис. 9.9). Для этой

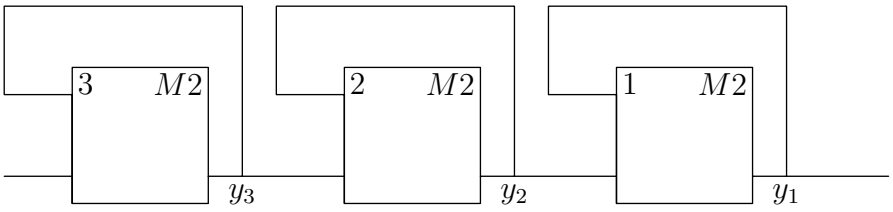


Рис. 9.9: Пример генератора с локальными обратными связями

схемы

$$L = \begin{pmatrix} 1 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

В этом случае матрица

$$D = L \oplus I = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

Нетрудно проверить, что $\mathrm{rank}(D) = n-1,\ \mathrm{rank}(\bar{D}) = n$, поэтому схема не имеет стабильных состояний. Матрица L будет разложимой, поскольку прибавив к ней единичную матрицу, получим треугольную матрицу, которая при возведении в произвольную степень останется треугольной матрицей. С другой стороны, для этой схемы не существует частично стабильных состояний. Действительно, для $i < n$ уравнение (9.8.26) принимает форму

$$y_i \oplus y_{i+1} = y_i.$$

Отсюда следует, что $y_{i+1} = 0$, поэтому из стабильности сигнала y_i следует стабильность сигнала y_{i+1}. В то же время, сигнал y_n не может быть постоянным, поскольку $y'_n = y_n \oplus 1$.

Рассмотрим пример схемы с частично стабильными состояниями. Пусть

$$L = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad D = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Легко видеть, что $\mathrm{rank}(D) = 3$. Аналогом уравнений (9.8.27) в этом случае являются уравнения

$$\lambda_i \epsilon_j = 0,\ i = 1,4; j = 2,3.$$

Это означает, что выполнены необходимые условия для существования состояния со стабильными битами в позициях 1 и 4. Для выяснения достаточности этих условий надо рассмотреть уравнения (9.8.26):

$$\lambda_i \mathbf{B} = y_i \oplus b_i,\ i = 1,4.$$

В данной ситуации эти уравнения имеют решения для любых значений $b_1, \ldots, b_4$. Выберем эти значения таким образом, чтобы было выполнено неравенство $rank(D) < rank(\bar{D})$. Достаточно положить их равными $1, 0, 0, 0$. В этом случае не будет стабильных состояний, но состояние $\langle 1, *, *, 1\rangle^T$ будет частично стабильным, поскольку не меняются биты в позициях 1 и 4.

9.9 Генератор случайных чисел на основе трехзначной логики

Увеличение производительности цифровых устройств является одной из основных задач современной схемотехники. Достигнутая к настоящему

времени тактовая частота близка к предельно возможной. Дальнейшее повышение производительности осуществляются либо за счет распараллеливания вычислений либо путем применения устройств, работающих в k-значной логике. Следует отметить, что на заре развития вычислительной техники существовали компьютеры работающие на основе трехзначной логики. Здесь, прежде всего, следует упомянуть Н.П. Брусенцова и его машину "Сетунь". Впоследствии интерес к таким устройствам упал в связи с производством стандартных двоичных чипов, реализующих все необходимые функции. В последнее время созданы простые физические устройства, реализующие трехзначную логику (см. [37]), что пробудило интерес к этим схемам и стимулирует разработку полезных физических приборов, работающих в трехзначной логике.

Ранее было дано математическое описание физических генераторов на основе двоичной логики. Оказалось, что аналогичный подход годится и для модели генератора, работающего в троичной логике. В его основе лежит специальная комбинационная схема трехзначной логики, работающая в режиме дребезжания. Такой генератор обладает рядом интересных свойств, поэтому дальнейшее развитие схемотехники позволит найти практическое применение для этих генераторов. В частности, они могут быть использованы для генерации криптографических ключей. Изложение данного пункта основано на работе [82].

9.9.1 Пример генератора

В качестве иллюстрации рассмотрим следующий пример схемы. Генератор построен по кольцевому принципу. Он состоит из одинаковых комбинационных схем (блоков), соединенных в кольцо. Каждый блок имеет два входа и один выход. Пример генератора из трех блоков представлен на Рис.9.10. Способ соединения произвольного числа блоков следует из

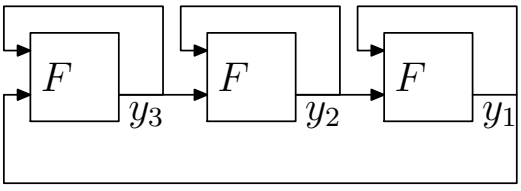


Рис. 9.10: Пример генератора тернарных последовательностей

этого рисунка очевидным образом. Здесь символом F обозначен упомянутый блок, а генерируемый сигнал снимается с выхода любого из блоков.

Сигнал является троичным, благодаря чему увеличивается производительность устройства по сравнению с двоичным аналогом, работающим с той же скоростью.

9.9.2 Математическая модель генератора

При создании указанных генераторов возникают следующие проблемы:

1. возможность перехода генератора в стабильное состояние;
2. обоснование нужных статистических свойств сгенерированных символов.

Ниже будет показано, что при надлежащем выборе блока F и способа соединения блоков между собой генератор не имеет стабильных состояний. Что касается статистических свойств генерируемой последовательности, то о них можно говорить лишь после того, как будут выбраны математическая модель функционирования отдельных блоков и способ съема сигнала. Естественными требованиями являются равномерность распределения выходного сигнала и независимость снимаемых сигналов. В этом случае появляется возможность преобразования выходного сигнала в сигнал с произвольным распределением. Общий подход к исследованию генератора, составленного из нелинейных блоков, представлен в предыдущей главе, однако применение трехзначной логики вносит некоторые упрощения в описание ситуации. Каждый из блоков реализует некоторую функцию $c = F(a, b)$, $a, b, c \in \{0, 1, 2\}$. При изменении входных сигналов блок срабатывает, реализуя функцию F. Относительно срабатывания блоков сделаны предположения аналогичные ограничениям, использованным при моделировании двоичных схем:

1. время срабатывания блока является случайной величиной с экспоненциальным распределением.
2. срабатывания отдельных блоков являются независимыми событиями, причем никакие два блока не могут сработать одновременно.

Ради простоты в дальнейшем будем предполагать, что параметр экспоненциального распределения равен 1. Перенумеруем блоки генератора числами от 1 до n. Состоянием генератора в момент времени t назовем вектор $\mathbf{S}(t) = \langle y_1, \ldots, y_n \rangle^T$, компонента которого y_k есть сигнал на выходе блока с номером k в момент времени t. В отличие от двоичного случая, $y_k \in \{0, 1, 2\}$. Если генератор содержит n блоков, то число его состояний

равно $m = 3^n$. Благодаря сделанным предположениям функционирование генератора описывается уравнениями Эрланга [83]. Система строится аналогично двоичному случаю, но для полноты изложения приведем вывод уравнений. Перенумеруем все состояния генератора числами от 1 до m. Пусть $\mathbf{S}_q$ — состояние с номером q, а $P_q(t)$ — вероятность того, что в момент времени t генератор находится в состоянии $\mathbf{S}_q$. Обозначим через $i_1, i_2, \ldots, i_r$ — все номера состояний, свои для каждого q, из которых можно попасть в состояние $\mathbf{S}_q$ в результате срабатывания только одного блока. Вероятность того, что через момент Δt генератор окажется в состоянии $\mathbf{S}_q$, складывается из вероятности того, что генератор находился в этом состоянии прежде и ни один из n блоков не сработал и из вероятностей перейти в состояние $\mathbf{S}_q$ из одного из состояний с номерами $i_1, i_2, \ldots, i_r$ в результате срабатывания только одного блока. Учитывая только слагаемые первой степени по Δt, получим, что вероятность срабатывания одного блока равна Δt, а вероятность того, что ни один блок на сработает равна $1 - n\Delta t$. Теперь

$$P_q(t + \Delta t) = (1 - n\Delta t)P_q(t) + \sum_{k=1}^{r} P_{i_k}\Delta t,$$

откуда, деля на Δt и переходя к пределу при $\Delta t \to 0$, получим

$$\frac{dP_q(t)}{dt} = -nP_N(t) + \sum_{k=1}^{r} P_{i_k}(t). \tag{9.9.28}$$

Функционирование системы полностью описывается с помощью решения системы (9.9.28), однако размер матрицы этой системы быстро растет с увеличением числа блоков, что делает затруднительным исследование особенностей поведения системы. Ниже будет показано, что используя дополнительные соображения, можно сделать выводы о свойствах решения, не решая саму систему.

9.9.3 Выбор функции F

При выводе уравнения (9.9.28) не делалось никаких предположений о виде функции F. В данном пункте будут приведены соображения по поводу выбора этой функции. Прежде всего, нужно гарантировать отсутствие стационарных состояний генератора. Предположим, что функция F обладает следующим свойством:

$$c = F(a, b), \quad \forall (a, b) \quad c \neq a, b. \tag{9.9.29}$$

Из условия (9.9.29) следует, что при срабатывании любого блока состояние генератора изменится при любом соединении блоков между собой. Таким образом, данное условие исключает наличие стационарных состояний. Перечислим все функции, обладающие свойством (9.9.29). Значение $F(a,b)$ определено однозначно, если $a \neq b$, поэтому $F(a,b) = F(b,a)$. Это означает, что достаточно определить функцию лишь для совпадающих аргументов. Если σ — произвольная перестановка чисел $0, 1, 2$, то функции $F(a,b)$ и $\sigma(F(\sigma(a), \sigma(b))$ будем считать неразличимыми. Действительно, для наших целей не играет роли, как будет обозначен тот или иной элемент троичной логики. Отсюда следует, что существуют лишь две существенно разные функции, обладающие свойством (9.9.29):

	$F(0,0)$	$F(1,1)$	$F(2,2)$
F_1	1	2	0
F_2	1	2	1

Изучаемый генератор предназначен для генерации последовательностей, в которых каждый из элементов появляется с одной и той же вероятностью. В этой связи далее в качестве функции F будем использовать только F_1. В дальнейшем будем применять символ F для обозначения этой функции.

Теперь надо определить способ соединения блоков между собой. Остается открытым вопрос о связи топологии соединения блоков со свойствами генератора. В данном пункте будет изучен лишь один из вариантов соединения, представленный на Рис.9.10. Достоинством указанной схемы является равноправность всех выходов блоков и одинаковая электрическая нагрузка на выходе каждого блока. Последнее условие, которое носит технологический характер, важно и с точки зрения адекватности математической модели. Время срабатывания физического блока может зависеть от того, на сколько входов будет подан сигнал. В этой связи, равноправность всех блоков в предложенной схеме становится существенным элементом. В дальнейшем будет показано, что схема обладает свойством «забывания» начального состояния, поэтому, в силу отмеченной симметрии, на выходе любого блока при снятии сигнала через достаточно большой интервал времени t_0 статистические свойства сигнала будут одними и теми же. Более того, функция F обладает тем свойством, что любое значение на выходе появляется одинаковое количество раз, когда аргументы пробегают всю область определения функции. Интуитивно ясно, что в силу этого обстоятельства на выходе каждого блока

генерируется сигнал с равномерным распределением. Далее будет доказана справедливость этого предположения.

9.9.4 Матрица переходов генератора

Для обоснования статистических свойств снимаемого сигнала необходимо изучить свойства матрицы A переходов генератора. Это матрица размера $m \times m$ $m = 3^n$. Напомним, что матрица состоит из нулей и единиц, причем $A[i|k] = 1$ тогда и только тогда, когда при срабатывании какого-либо блока генератор переходит из состояния с номером i в состояние с номером k. Рассмотрим произвольное состояние

$$\mathbf{S} = \langle y_1, \ldots, y_k, \ldots, y_N \rangle^T. \tag{9.9.30}$$

В силу свойства (9.9.29), в результате срабатывания любого из n блоков состояние генератора изменится, причем все получившиеся состояния будут разными. Это означает, что каждая строка матрицы A имеет ровно n единиц.

Предложение 5 *Состояния (9.9.30), в которых все сигналы равны между собой, при указанных выборе функции F и способе соединения блоков недостижимы из других состояний генератора.*

Доказательство

Рассмотрим состояние $\mathbf{S_0} = \langle 0, \ldots, 0 \rangle^T$. Если из некоторого состояния $\mathbf{S_1}$ возможен переход в состояние $\mathbf{S_0}$ в результате срабатывания одного блока, то все компоненты вектора $\mathbf{S_1}$, кроме одной, равны 0. Легко видеть, что после срабатывания любого блока, в силу (9.9.29), переход в состояние $\mathbf{S_0}$ невозможен. Поскольку в рассматриваемой схеме все троичные значения равноправны, аналогичные утверждения справедливы для векторов $\langle 1, \ldots, 1 \rangle^T$ и $\langle 2, \ldots, 2 \rangle^T$.

Из доказанного утверждения следует, что столбцы с номерами, отвечающими трем недостижимым состояниям, будут нулевыми. Удалим из A эти три строки и три столбца с указанными номерами и обозначим через A' получившуюся матрицу. Это квадратная матрица с неотрицательными элементами, и все утверждения, относящиеся к таким матрицам, представленные выше, остаются справедливыми и для этой матрицы.

Свойства генерируемой последовательности базируются на следующей теореме.

Теорема 2 *Матрица A' является неразложимой.*

Доказательство

Под множеством состояний будем понимать множество всех состояний генератора, кроме трех отмеченных недостижимых состояний. Достаточно доказать, что из любого состояния можно перейти в любое другое. Пусть $\mathbf{S_1} = \langle 1, 0, \ldots, 0\rangle^T$. Покажем, что из $\mathbf{S_1}$ можно перейти в любое другое состояние через несколько шагов, свое для каждого состояния. После двукратного срабатывания второго блока генератор перейдет в состояние $\langle 1, 1, 0, \ldots, 0\rangle^T$, а затем – в состояние $\langle 1, 2, 0, \ldots, 0\rangle^T$. Итак, доказано, что из $\mathbf{S_1}$ можно перейти в состояние $\langle 1, a, 0, \ldots, 0\rangle^T$, где a – любой элемент множества $L = \{0, 1, 2\}$. При срабатывании первого блока переходим из состояния $\mathbf{S_1}$ в состояние $\langle 2, 0, 0, \ldots, 0\rangle^T$. После этого, как и выше, доказывается, что достигается любое состояние $\langle 2, a, 0, \ldots, 0\rangle$, где $a \in L$. Предположим, что уже доказана достижимость из состояния $\mathbf{S_1}$ любого состояния вида $\langle 1, a_2, a_3, \ldots, a_k, 0, \ldots, 0\rangle^T$ или $\langle 2, a_2, a_3, \ldots, a_k, 0, \ldots, 0\rangle^T$, где $a_2, a_2, \ldots, a_k$ – любые элементы множества L. Если $k < n-1$, то при срабатывании блока с номером $k+1$ из состояния $\langle 1, a_2,$
$a_3, \ldots, a_k, 0, 0, \ldots, 0\rangle$ переходим в состояние $\langle 1, a_2, a_3, \ldots, a_k, 1, 0, \ldots, 0\rangle$, а затем — в $\langle 1, a_2, a_3, \ldots, a_k, 2, 0, \ldots, 0\rangle^T$. Если $k = n - 1$, то при срабатывании блока с номером n из состояния $\langle 1, a_2, a_3, \ldots, a_k, 0\rangle^T$ переходим в состояние $\langle 1, a_2, a_3, \ldots, a_k, 2\rangle^T$, а из состояния $\langle 2, a_3, \ldots, a_k, 0\rangle^T$ – в $\langle 2, a_2, a_3, \ldots, a_k, 1\rangle^T$. Таким образом, доказана достижимость состояний вида $\langle a_1, \ldots, a_n\rangle^T$, где $a_1 \neq a_n$ и $a_1 \neq 0$. После срабатывания первого блока переходим из состояния $\langle 1, 2, a_3, \ldots, a_n\rangle^T$ в состояние $\langle 0, 2, a_3, \ldots, a_n, \rangle^T$, а из состояния $\langle 2, 1, a_3, \ldots, a_n\rangle^T$ в состояние $\langle 0, 1, a_3, \ldots, a_n, \rangle^T$. Это означает, что из состояния $\mathbf{S_1}$ достижимо любое состояние вида $\langle 0, a_2, a_3, \ldots, a_n, \rangle^T$, где $a_2 \neq 0$. Продолжая очевидным образом, получаем, что достижимо любое состояние вида $\langle 0, 0, \ldots, a_k, \ldots, a_n, \rangle^T$, где $a_k \neq 0, k \leq n$. Наконец, покажем достижимость состояния вида $\langle 1, \ldots, 1, b_p, \ldots, b_{n-1}, 1\rangle^T$, где $b_p \neq 1$. Согласно предположению, достижимо состояние $\langle 2, 0, \ldots, 0, b_p, \ldots, b_{n-1}, 1\rangle^T$. После последовательного срабатывания блоков с номерами $1, 2, \ldots, p-1$ получим состояние $\langle 1, \ldots, 1, b_p, \ldots, b_{n-1}, 1\rangle^T$, поскольку $b_p = 0$ или $b_p = 2$. Точно также доказывается достижимость состояния вида $\langle 2, \ldots, 2, b_p, \ldots, b_{n-1}, 2\rangle^T$, где $b_p \neq 2$. Достижимость состояния вила $\langle 0, \ldots, , b_p, \ldots, b_{n-1}, 0\rangle^T$, где $b_p \neq 0$, была доказана выше.

Теперь покажем, что из любого состояния $\mathbf{S}$ можно перейти в состояние $\mathbf{S_1}$. При срабатывании первого блока происходит переход из состояния $\langle 0, a_2, \ldots, a_n\rangle^T$ в состояние $\langle b, a_2, \ldots, a_n\rangle^T$, где $b = 1$ или $b = 2$. Это

означает, что можем ограничиться состояниями, начинающимися с 1 или 2. Из состояния $\langle 1,0,\ldots,0,1\rangle^T$ после двукратного срабатывания блока с номером n получаем состояния $\langle 1,0,\ldots,0,2\rangle$, $\langle 1,0,\ldots,0,0\rangle^T$. Из состояния $\langle 2,0,\ldots,0,0\rangle^T$ после срабатывания блока с номером n получается состояние $\langle 2,0,\ldots,0,1\rangle^T$, а после срабатывания первого блока получаем $\langle 1,0,\ldots,0,1\rangle^T$. Из состояния $\langle 2,0,\ldots,0,2\rangle^T$ после срабатывания блока с номером n получается состояние $\langle 2,0,\ldots,0,0\rangle^T$.

Это означает, что состояние $\mathbf{S_1}$ достижимо из состояний вида $\langle 1,0,\ldots,0,a\rangle$, $\langle 2,0,\ldots,0,a\rangle^T$ для любых $a \in L$. Пусть уже доказана достижимость $\mathbf{S_1}$ из состояний вида $\langle 1,0,\ldots,0,b_p,\ldots,b_n\rangle^T$ для любых $b_i \in L$. Рассмотрим произвольное состояние вида $\langle 1,0,\ldots,0,b_{p-1},b_p,\ldots,b_N\rangle^T$, $b_{p-1} \neq 0$. Возможны следующие наборы значений b_{p-1}, b_p : (1,2), (2,1), (2,2), когда после срабатывания блока с номером $p-1$ получаем 0 в позиции $p-1$; (1,1), когда после двукратного срабатывания блока с номером $p-1$ получаем 0 в позиции $p-1$; (2,0), (1,0) – последняя ситуация требует дополнительного рассмотрения. Рассмотрим состояние $\langle 1,0,\ldots,0,1,0,b_{p+1},\ldots,b_N\rangle^T$. При срабатывании блока с номером p в этой позиции появится 1 или 2, что сводит эту ситуацию к предыдущему случаю. Для завершения доказательства достаточно поменять местами значения 1 и 2.

9.9.5 Статистические свойства генерируемой последовательности

Введем обозначение $B = A^T$. Согласно определению матрицы A, каждая строка матрицы B с номером i содержит 1 в позиции j тогда и только тогда, когда возможен переход из состояния с номером j в состояние с номером i в результате срабатывания одного блока. Рассмотрим более подробно систему уравнений (9.9.28). Положим $\mathbf{P}(t) = (P_1(t),\ldots,P_m(t))^T$. Указанная система может быть переписана в виде

$$\frac{d\mathbf{P}(t)}{dt} = (B - n\cdot I)\mathbf{P}(t) = D\mathbf{P}(t). \tag{9.9.31}$$

Решение данной системы имеет вид

$$\mathbf{P}(t) = \exp(Dt)\mathbf{E},$$

где $\mathbf{E}$ – произвольный стохастический вектор, определяющий начальное состояние. Матрица B имеет три нулевых строки; пусть это строки с номерами 1, 2, 3. Из (9.9.31) следует, что

$$P_k(t) = e_k \exp(-nt), \quad k \in \{1,2,3\}, \tag{9.9.32}$$

где $0 \le e_k \le 1$. Отсюда вытекает, что $P_k(t) \to 0$ при $t \to \infty$, $k \in \{1,2,3\}$. Исключим из векторов компоненты с индексами 1, 2, 3, а из матриц — строки и столбцы с этими номерами. В результате получим векторы $\mathbf{E}'$, $\mathbf{P}'(t)$, $D' = A'^T - n \cdot I'$, а решение системы (9.9.31) сведется к вычислению

$$\mathbf{P}'(t) = \exp(D't)\mathbf{E}'. \qquad (9.9.33)$$

Теорема 3 *Пусть* $C(t) = \exp(Dt)$. *Тогда* $C(t) \to C_0$ *при* $t \to \infty$, *где* C_0 – *матрица с одинаковыми столбцами, равными стохастическому вектору* $\mathbf{d}$, *такому, что* $\mathbf{d} = \langle 0,0,0,\mathbf{d}'\rangle^T$, $A'^T\mathbf{d}' = n\mathbf{d}'$.

Доказательство

Из (9.9.32) вытекает, что первые три строки матрицы C_0 нулевые. Сумма элементов в каждой строке матрицы A равна n и при выбранной нумерации состояний её первые три столбца нулевые. Таким образом, матрица A'/n стохастическая, ее максимальное характеристическое число равно 1, все остальные характеристические числа имеют модули, не превосходящие 1 (см. [71] с.200), а из неразложимости этой матрицы вытекает, что 1 является простым корнем. Другими словами, если $r_1, \ldots, r_{m-3}$ — все характеристические числа матрицы A', и r_1 – максимальный по модулю вещественный корень, то $r_1 = n$, $Re(r_i) < n$, $i > 1$. По определению $D' = A'^T - n \cdot I'$, поэтому для характеристических чисел q_j этой матрицы выполнены условия $q_1 = 0$, $Re(q_i) < 0$, $i = 2, \ldots, m-3$. Характеристические числа матрицы $C'(t) = \exp(D't)$ равны $\exp(q_i t)$, поэтому модули всех характеристических чисел, кроме первого, меньше 1. Отсюда следует существование предела $C'_0 = \lim_{t\to\infty} C'(t)$, и матрица C'_0 имеет ранг 1. Если $A'^T\mathbf{d}' = n\mathbf{d}'$, то $D'\mathbf{d}'$ — нулевой вектор, а из представления матрицы $C'(t)$ в виде ряда вытекает, что $C'(t)\mathbf{d}' = \mathbf{d}'$ для любого t. Это означает, что $C'_0\mathbf{d}' = \mathbf{d}'$. Далее, $\langle 1,1,\ldots,1\rangle A'^T = n\langle 1,1,\ldots,1\rangle$, поэтому $\langle 1,1,\ldots,1\rangle D'$ — нулевой вектор и $\langle 1,1,\ldots,1\rangle C'_0 = \langle 1,1,\ldots,1\rangle$, т. е. сумма элементов в каждом столбце этой матрицы равна 1. Это замечание завершает доказательство теоремы

Вектор $\mathbf{d}'$ задает финальное распределение вероятностей состояний генератора. Из Теоремы 3 следует независимость этого распределения от начального состояния, а финальные вероятности появления 0, 1 и 2 на выходе любого блока равны $1/3$. Последнее утверждение вытекает из симметричности схемы. В равновероятности появления каждого из чисел 0, 1 и 2 на выходе можно убедиться непосредственно. Например, чтобы найти вероятность появления 1 на выходе первого блока, надо выписать все состояния, у которых в троичной кодировке номера в первой пози-

ции стоит 1. Если число блоков равно 2, то такими состояниями будут состояния с номерами 3,4 и 5. После этого суммируют финальные вероятности каждого из этих состояний. Следует, однако, заметить, что отсюда нельзя заключить, что финальные вероятности каждого из состояний генератора совпадают между собой. Они будут, вообще говоря, разными.

9.9.6 Результаты численных экспериментов

При практическом использовании разработанного генератора возникает вопрос о скорости сходимости решения уравнения (9.9.31) к финальному вектору в зависимости от числа n блоков в схеме. В качестве меры близости было выбрано средне-квадратическое отклонение δ финального вектора — собственного вектора матрицы D, отвечающего собственному значению 0, от первого столбца матрицы $\exp(Dt)$. Результаты экспериментов помещены в следующую таблицу:

Таблица Зависимость δ от n и t

$n\backslash t$	2	4	6	8	10
2	0.0095249	0.0001857	0.0000034	6.288D-08	1.141D-09
3	0.0026750	0.0000398	0.0000005	7.596D-09	1.102D-10
4	0.0014727	0.0000404	0.0000011	3.592D-08	1.113D-09
5	0.0004785	0.0000077	8.981D-08	9.856D-10	3.056D-11
6	0.0001991	0.0000055	0.0000002	6.546D-09	2.868D-10

Из приведенных результатов следует, что при любом n схема, практически, забывает свое начальное состояние при $t \geq 5$. Еще раз подчеркнем, что расчет велся для случая, когда параметр экспоненциального распределения равен 1. Для произвольного значения a этого параметра нужно заменить t на ta.

Литература

[1] Agrawal M. *PRIMES is in P* / M.Agrawal, N.Kayal, N.Saxena.– Annals of Mathematics.– 2004, v.160, p. 781–793.

[2] Atkin A. *Prime sieves using binary quadratic forms*/ A. Atkin, D. Bernstein.– http://cr.yp.to/papers/primesieves–19990826.pdf

[3] Berstein D. *ECM using Edwards curves*/ D. Berstein, P. Birkner, T.Lange, C. Peters.–2008, p.1–40 http://eecm.cr.yp.to/eecm-20100616.pdf

[4] Berstein D. *Faster addition and doubling on elliptic curves.*/ D. Berstein, T.Lange. in AsiaCrypt'2007, p.29–50

[5] Berstein D. *Explicit-formulas Database.*/ D. Berstein, T.Lange. 2007 http:// hyperelliptic.org/EFD

[6] Berstein D. *Starfish on Strike*/ D. Berstein, P. Birkner, T.Lange, C. Peters.–LATINCRYPT 2010, edited by Michel Abdalla and Paulo S. L. M. Barreto. Lecture Notes in Computer Science 6212. Springer, 2010, p.61–80

[7] Boldyreva A. *Efficient Thresholf Signature, Multisignature and Blind Signature Schemes based on Diffie–Hellman–Group Signature Scheme.* Crypto'2003, Lect.Not.Comp.Sci., p.31–46

[8] Boneh D.,Franklin M. *Identity based encryption from the Weil pairing.* In J.Killan, editor, Proceeding of Crypto'2001, volume 2139, Lect.Notes in Comp.Sci., 2001, p.213–229

[9] Brent R.P. *Some integer factorization algorithms using elliptic curves*/ R.P. Brent.– Austral.Comput.Sci.Comm, 1986, v.8, p. 149–163.

[10] Buhler J.P. *Factoring integers with the number field sieve* / J. P. Buhler, H. W. Lenstra, C. Pomerance.– in The Developement of the Number Field Sieve, Springer–Verlag, Berlin, Germany, 1993, p. 50–94.

[11] Chaum D. *Zero-knowledge undeniable signatures.* In I.Damgard, editor, Advances in Cryptology–Crypto'90, Lect.Not.Comp.Sci., v.740, 1992, p.89-105

[12] Cocks C. *An identity based encryption scheme based on quadratic residues.* Cryptography and Coding, 2001.

[13] Cohen H. *A course in computational algebraic number theory* / H. Cohen.– Springer–Verlag, Berlin, 1993, 545 p.

[14] Crandall R. *The prime numbers: a computational perspertive* / R. Crandall, C. Pomerance.– sec.ed. Springer–Verlag, Berlin, 2005, 604 p.

[15] Dunham W. *Euler : The Master of Us All.* Mathematical Association of America, 1999, 185 p.

[16] Edwards H.M. *A normal form for elliptic curves.*/ H.M. Edwards.–Bull. Amer. Math. Soc. 44 (2007), p. 393-422

[17] Elkenbracht-Huising M. *An implementation of the Number Field Sieve* / M. Elkenbracht-Huising.– Experimental Mathematics, 1996, v.5, p. 231—-253.

[18] Gardner M. *A new kind of cipher that would take millions years to break* / M. Gardner.– Sci. Amer. 1977, p. 120–124.

[19] Golomb S.W *Shift register sequences* San Francisco, Holden-Day — 1967 — 115 p.

[20] Granville A.*Smooth numbers: Computational number theory and beyond*/ A. Granville.– Proc. of MSRI workshop, 2004, 268–363

[21] Hackmann P. *Elementary Number Theory* / P. Hackmann.– HHH Publ, 2007, 411 p.

[22] Jaeschke G. *On Strong Pseudoprimes to Several Bases* / G/ Jaeschke. - Mathematics of Computation, v.61, 1993, p.915-926

[23] Joux A. *A one round protocol for tripartie Diffie-Hellman.* / A. Joux.– Algorithmic Number Theory: 4-th International Symposium, ANT–IV, Lecture Notes in Computer Science, v.1838(2000), Springer–Verlag, p. 385–393.

[24] Joux A. *The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems.* Proceedings of the 5th International Symposium on Algorithmic Number Theory, Springer-Verlag London, 2002, p.20–32

[25] Lenstra H.W. *Factoring integers with elliptic curves* / H.W. Lenstra.– Ann.Math. v.126 (1987), p. 649–674.

[26] Lenstra A. *The Development of the Number Field Sieve* / A. Lenstra and H. Lenstra (eds.).– Lect.Not.in Math.**1554**, Springer–Verlag, Berlin, 1993, 139 p.

[27] Longa P.*Fast Point Arithmetic for Elliptic Curve Cryptography*/ P. Longa.– Presentation at CliCC, University of Ottawa, Ottawa, Canada, 2006.

[28] Longa P. *ECC Point Arithmetic Formulae (EPAF): Jacobian coordinates*/ P. Longa, C. Gebotus. In Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES 2010), 2010.

[29] Menezes A. *Reducing Elliptic Curve Logarithms to a Finite Field* / A. Menezes, T. Okamoto, S. Vanstone.– IEEE Trans. Info. Theory, v.39, 1993, p. 1639–1646.

[30] Menezes A. *Elliptic Curve Public Key Cryptosystems* / A. Menezes.– 1993, 144 p.

[31] Montgomery P.L. *Speeding the Pollard and Elliptic Curve Methods of Factorization.*/P.L. Montgomery.– Mathematics of Computation, v.48, iss.177, 1987, p.234–264.

[32] Montgomery P.L. *An FFT-extension of the Elliptic Curve Method of Factirization* / P.L. Montgomery.– Doctoral Dissertation, 1992, Univ.Calif. USA, 118 p.

[33] Pollard J.M. *Theorems on factorization and primality testing* / J.M. Pollard. – Proc.Cambridge Phil.Society. 1974, v.76, p. 521-578.

[34] Pomerance C. *Smooth Numbers and the Quadratic Sieve* / C. Pomerance. – MSRI publications, **v.44** – 2008, p. 69–82.

[35] Shoup V. *A Computational Introduction to Number Theory and Algebra*/ V. Shoup. – Cambridge University Press, Sec.Edition, 2005, 600 p. http://shoup.net/ntb/

[36] SciLab *Free open source software for numerical computation* /http://www.scilab.org

[37] Sheng Lin, Yong-Bin Kim, Lombardi F. CNTFET-Based Design of Ternary Logic Gates and Arithmetic Circuits //IEEE Transactions on Nanotechnology, March. 2011, vol.10, № 2, С. 217-225

[38] Venturi D. *Lecture Notes on Algorithmic Number Theory.*/ D. Venturi. – Springer-Verlag, New-York, Berlin, 2009, 217 p.

[39] Washington L. *Elliptic Curves Number Theory and Cryptography* /L. Washington. – Series Discrete Mathematics and Its Applications, Chapman & Hall/CRC,second ed. 2008, 524 p.

[40] Аграновский А.В. *Практическая криптография: алгоритмы и их программирование* / А.В. Аграновский, Р.А. Хади.– М.: Солон-Пресс, 2009, 256 с.

[41] Айерленд К. *Классическое введение в современную теорию чисел.* / К. Айерленд, М. Роузен. – М.: Мир, 1987, 428 с.

[42] Акритас А. *Основы компьютерной алгебры и приложениями.* / А. Акритас. – М.: Мир, 1994, 544 с.

[43] *Алгебраическая теория автоматов, языков и полугрупп*/ Под редакцией М.А. Арбиба — М.: Статистика — 1975 — 334 с.

[44] Беллман Р.*Введение в теорию матриц*/Р.Беллман – М.: Наука, – 1969. – 367 с.

[45] Берленкэмп Э.*Алгебраическая теория кодирования.* /Э.Берленкэмп. – М.: Мир, – 1971. – 478 с.

[46] Богопольский О.В. *Алгоритмическая теория чисел и элементы криптографии.* / О.В. Богопольский.– Спецкурс для студентов НГУ, Новосибирск, 2005, 35 с. / http://math.nsc.ru/ bogopolski/Articles/SpezkNumber.pdf

[47] Болотов А.А. *Элементарное введение в эллиптическую криптографию: протоколы криптографии на эллиптических кривых.* / А.А. Болотов, С.Б. Гашков, А.Б. Фролов. – М.:КомКнига, 2004, 280 с.

[48] Болотов А.А. *Алгоритмические основы эллиптической криптографии.* / А.А. Болотов, С.Б. Гашков, А.Б. Фролов, Часовских А.А.. – М.:РГСУ, 2004, 499 с.

[49] Боревич З.И. *Теория чисел.* / З.И. Боревич, И.Р. Шафаревич. – 3-е издание, М.: Наука, 1985, 504 с.

[50] Ван дер Варден Б.Л. *Алгебра.*/ Б.Л.ван дер Варден. – изд.2, М.: Наука, 1979, 623 с.

[51] Василенко О.Н. *Теоретико-числовые алгоритмы в криптографии*/ О.Н. Василенко. – МЦНМО, 2003, 326 с.

[52] Вельщенбах М. *Криптография на C и C++ в действии: учебное пособие* / М. Вельщенбах. – М.: Триумф, 2008, 464 с.

[53] Гатмахер Ф.Р. *Теория матриц.*/ Ф.Р. Гатмахер – М.: Наука. – 1967. – 575 с.

[54] Гилл А *Линейные последовательностные машины*/А. Гилл — М.:Наука — 1974 — 287 с.

[55] Ермаков С.М. *Курс статистического моделирования*/С.М. Ермаков, Г.А. Михайлов — М.:Наука — 1976 — 318 с.

[56] Захаров В.М. *Вычисления в конечных полях: уч.–метод. пособие* / В.М. Захаров, Б.Ф. Эминов. – Казань: КГТУ им. А.Н.Туполева, 2010, 132 с.

[57] Ишмухаметов Ш.Т. *Методы факторизации натуральных чисел*/ Ш.Т.Ишмухаметов. – Казань, 2012, 189 с.

[58] Кнут Д. *Искусство программирования для ЭВМ. т.2 Получисленные алгоритмы*/Д. Кнут — М.:Мир — 1977 — 724 с.

[59] Коблиц Н. *Курс теории чисел и криптографии* / Н. Коблиц. – М.: ТВП, 2001, 260 с.

[60] Корешков Н.А. *Теория чисел.*/Н.А. Корешков. – Уч.-мет. пособие, Казань, КФУ, 2010, 35 с.

[61] Кормен Т. *Алгоритмы: построение и анализ* /Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО, 1999.

[62] Крамер Г. *Математические методы статистики* /Г. Крамер — М.:Мир — 1975 — 648 с.

[63] Кузнецов В.М. *Марковская модель цифрового стохастического генератора*/В.М. Кузнецов, В.М. Песошин, Е.Л. Столов — Автоматика и Телемеханика — 2008 – № 9, с. 62-68

[64] Кузнецов В.М. *Марковская модель цифрового стохастического генератора*/В.М. Кузнецов, В.М. Песошин, Е.Л. Столов *Стабильные состояния асинхронного генератора.* – Ученые записки казанского государственного университета, –2010 – Том. 152, Книга 1. Серия - Физико-математические науки, с.174-180

[65] Кунегин С.В. *Системы передачи информации.* С.В. Кунегин.- Курс лекций. М., в/ч 33965, 1997, 317 с. http://kunegin.com/ref/sod_lec.htm

[66] Лазарева С.В. *Математические основы криптологии: тесты простоты и факторизация.* / С.В. Лазарева, А.А. Овчинников. Учебное пособие, Санкт-Петербург, СПбГУАП, 2006, 65 с.

[67] Латыпов Р.Х.*Периодические последовательности, порождаемые регистр сдвига с нелинейными обратными связями*/ Р.Х. Латыпов. Изв. Вузов. Математика, №5, 1989, С.5-13

[68] Лидл Р. *Конечные поля*/Р. Лидл,Г. Нидеррайтер.– Т. 1, 2. М.: Мир, 1988, 428 с.

[69] Мак-Вильямс Ф.Дж., Слоэн Н.Дж.А. *Теория кодов, исправляющих ошибки.*.– Ф.Дж. Мак-Вильямс, Н.Дж.А. Слоэн.– М. - Связь, 1979, 372 с.

[70] Максимов В.М.*О сходимости неоднородных дважды стохастических цепей Маркова*/В.М. Максимов —ТВП, т.15, вып. 4 — 1970 — с.622-636

[71] Маркус М., Минк Х.*Обзор по теории матриц и матричных неравенств*/М. Маркус , Х. Минк – М.: Наука, – 1972. – 232 с.

[72] Молдовян Н.А. *Криптография. От примитивов к синтезу алгоритмов* / Н.А.Молдовян, А.А. Молдовян,М.А. Еремеев. – БХВ-Петербург, 2004, 446 с.

[73] Морелос-Сарагоса Р. *Искусство помехоустойчивого кодирования - методы, алгоритмы, применение.*/ Р. Морелос-Сарагоса.– Москва, Техносфера, 2006, 312 с.

[74] Нестеренко Ю.В. *Теория чисел*/ Ю.В. Нестеренко. – Москва, Изд.Центр Академия, 2008, 273 с.

[75] Отнес Р.*Прикладной анализ временных рядов* /Р. Отнес, Л. Эноксон — М.:Мир — 1982 — 428 с.

[76] Песошин В.А. *Цифровые генераторы случайных сигналов для защиты информационных средств телекоммуникации* /В.А Песошин, В.М. Кузнецов, Н.Н. Сергеев. – Вопросы радиоэлектроники. Серия ЭВТ. Вып.4 —1993 — с.95-113.

[77] Песошин В.А. *Генераторы псевдослучайных и сдучайных чисел на регистрах сдвига* /В.А. Песошин, В.М. Кузнецов — Казань: КГТУ им. А.Н.Туполева — 2007 – 296 с.

[78] А.Г. Ростовцев, Е.Б. Маховенко. Теоретическая криптография, Профессионал, Санкт-Петербург, 2005, 479 с.

[79] Сизый С.В. *Лекции по теории чисел: учебное пособие для математических специальностей* / С.В. Сизый.– Екатеринбург, УрГУ, 1999, 136 с.

[80] Столов Е.Л. *Методы компактного тестирования цифровых устройств* / Е.Л. Столов — Казань: Казанский государственный университет — 1993—115 с.

[81] Столов Е.Л. *Генератор случайных чисел составленный из нелинейных асинхронных элементов*/ Е.Л. Столов. Исследования по прикладной математике, вып. 26, Институт информатики КГУ — Казань, — 2006 – с.101-106.

[82] Столов Е.Л. *Математическая модель генератора случайных чисел на основе трехзначной логики*/Е.Л. Столов – Прикладная дискретная математика – 2012 – № 2, с 43-49

[83] Хинчин А.Я. *Работы по математической теории массового обслуживания.*/А.Я. Хинчин – М.: Фзматгиз. – 1963. - 235 с.

[84] Чандрасекхаран К. *Введение в аналитическую теорию чисел*/ К. Чандрасекхаран.–М.– Мир, 1974, 187 с.

[85] Черемушкин А.В. *Лекции по арифметическим функциям в криптографии* / А.В. Черемушкин.– М.: МЦНМО, 2002, 103 с.

[86]) Шаньгин Ф.Ф. *Защита компьютерной информации: эффективные методы и средства* /Ф.Ф. Шаньгин.– М.:ДМК, 2008, 542 с.

[87]) Шеннон К. *Работы по теории информации и кибернетике.*/К. Шеннон.– М.: Изд-во иностранной литературы, 1963. — 830 с.