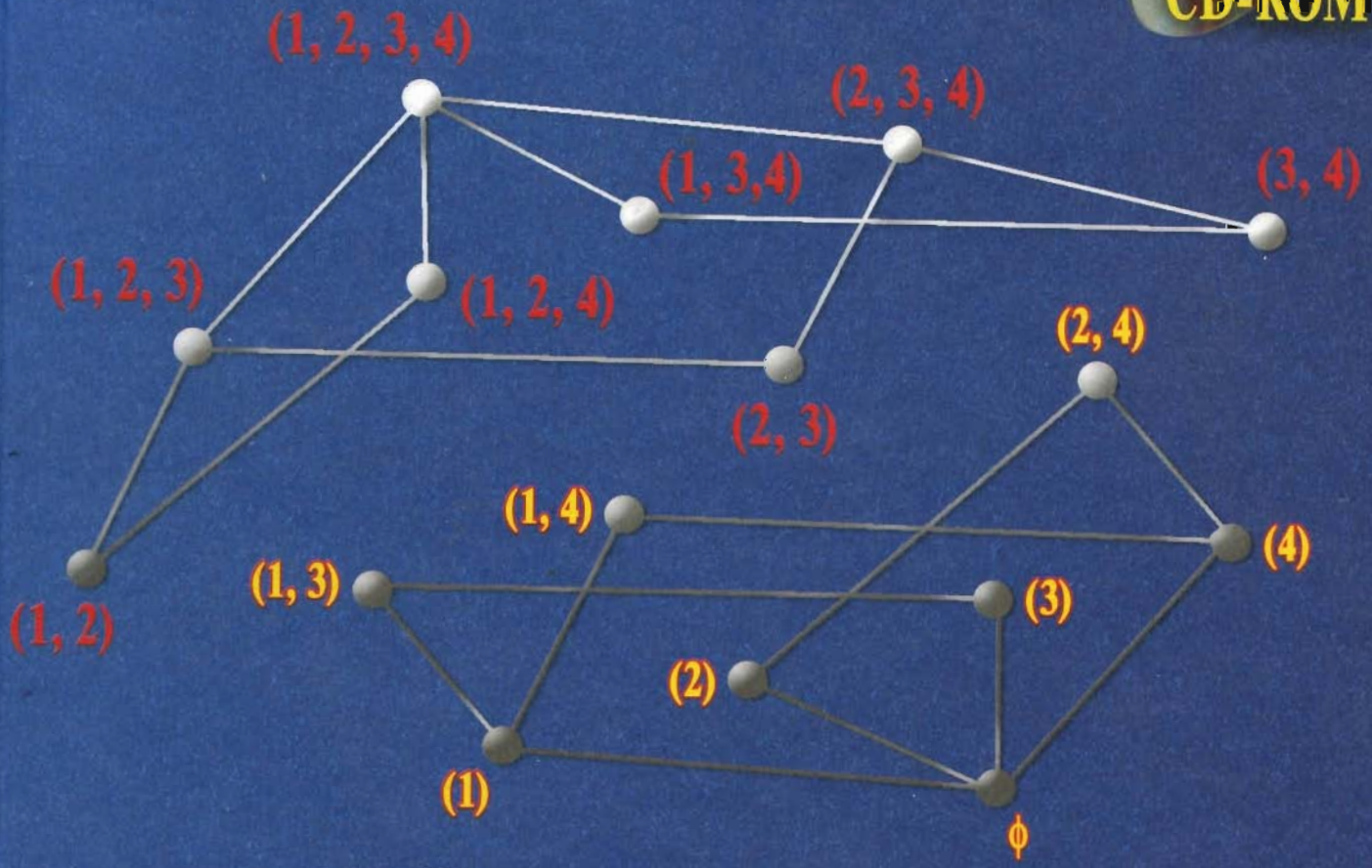
Х.К.А. ван Тилборг

ОСНОВЫ КРИПТОЛОГИИ

ПРОФЕССИОНАЛЬНОЕ РУКОВОДСТВО И ИНТЕРАКТИВНЫЙ УЧЕБНИК

CD-ROM



Издательство «МИР»

ОСНОВЫ КРИПТОЛОГИИ.
Профессиональное руководство и интерактивный учебник

The Kluwer International Series
in Engineering and Computer Science

FUNDAMENTALS OF CRYPTOLOGY

A Professional Reference and Interactive Tutorial

by

Henk C.A. van Tilborg
Eindhoven University of Technology
The Netherlands

KLUWER ACADEMIC PUBLISHERS
Boston / Dordrecht / London

Х.К.А. ван ТИЛБОРГ

ОСНОВЫ КРИПТОЛОГИИ

Профессиональное руководство и интерактивный учебник

Перевод с английского
Д.С. Ананичева и *И.О. Корякова*

под редакцией
И.О. Корякова

Москва «Мир» 2006

УДК 519.6
ББК 22.1
Т39

Тилборг ван Х.К.А.

Т39 Основы криптологии. Профессиональное руководство и интерактивный учебник. — М.: Мир, 2006, стр. 471, ил.

ISBN 5-03-003639-3

Книга голландского криптолога посвящена современнымм аспектам криптографии и криптоанализа. Среди них можно выделить три главных направления — традиционные (симметрические) криптосистемы, системы с публичными ключами и криптографические протоколы. Основные результаты снабжены доказательствами. Главной же особенностью служат многочисленные примеры, созданные на базе известного пакета "Mathematica" компьютерной алгебры. К книге приложен CD ROM, позволяющий (при наличии пакета "Mathematica") модифицировать примеры, в частности, увеличивая значения параметров. Это — первая столь многоплановая учебная книга по криптографии на русском языке.

Книга, в первую очередь, адресована математикам, инженерам и студентам, специализирующимся в области защиты информации. Но она окажется интересной и для более широкого круга читателей, чему, в частности, могут способствовать детальные приложения, посвященные теории чисел и конечным полям, делающие книгу достаточно замкнутой в себе.

УДК 519.6
ББК 22.1

Издание осуществлено при поддержке Российского фонда фундаментальных исследований по проекту 02-01-14009

Редакция литературы по математическим наукам

ISBN 5-03-003639-3(русск.)
ISBN 0-7923-8675-2 (англ.)

Предисловие редактора перевода

Криптология, объединяющая криптографию и криптоанализ, долгое время была секретной областью знаний в Советском Союзе. Преподавать криптографию имел право только ИКСИ — институт криптографии, связи и информатики (бывший технический факультет академии КГБ–ФСБ). Положение в корне изменилось лишь в 1990-х годах, когда стало невозможно игнорировать экстенсивное развитие компьютерных сетей, вызвавшее потребность банков, промышленных и коммерческих фирм, частных лиц в квалифицированной защите информации, чтобы предотвращать получение конфиденциальных данных неправомочными лицами и организациями, а также исключать фальсификацию и злонамеренное искажение данных. Именно в те годы некоторым гражданским вузам разрешили принять участие в подготовке криптологов, и были введены новые математические специальности в области защиты информации и компьютерной безопасности.

Одной из основных проблем, связанных с внедрением новых специальностей, была и остается проблема учебной и справочной литературы. Справочная литература (наподобие фундаментальной, хотя и слегка устаревшей, книги [MeOoV97]) в России практически отсутствует. Учебных же книг мало, и зачастую они не удовлетворяют обычным требованиям, предъявляемым к учебной математической литературе. Эти обстоятельства и побудили нас заключить контракт с издательством "Мир" о переводе книги Хенка К.А. ван Тилборга.

На наш взгляд, представляемая читателю книга, основанная на лекциях, читанных автором в США и Нидерландах, отчасти заполняет пробелы в имеющейся учебной литературе и может послужить хорошим дополнением к ней. Отметим, что ван Тилборг доказывает большинство приводимых утверждений, оставляя без подробного обоснования лишь наиболее сложные факты. Главный акцент падает на разъяснение ключевых идей, лежащих в основе криптографических алгоритмов и методов их потенциального криптоанализа. Но не последняя роль отводится и частным деталям, внимание к которым привлекают и многочисленные примеры алгоритмов и конкретных вычислений.

Важнейшая особенность книги — перманентное использование в примерах псевдокода системы компьютерной алгебры "Mathematica", которая была разработана чуть более 15 лет назад и быстро выдвинулась на ведущие позиции в области компьютерных вычислений. "Псевдо" — поскольку далеко не всегда выдерживается каноническая нотация системы: в целях облегчения восприятия часто употребляются стандартные математические обозначения. Это касается, например, записи сте-

пеней, "длинных" сумм и произведений (с символами $\sum$ и $\prod$) и т. п. Тем не менее, читатель вполне может изучить по этим примерам основы работы с пакетом "Mathematica". Более того, книга сопровождается компакт-диском с электронным вариантом текста. Поэтому, имея в своем распоряжении какую-либо версию пакета "Mathematica" и упомянутый CD ROM, можно легко модифицировать параметры примеров, углубляя тем самым понимание как методов криптологии, так и самой системы "Mathematica".

Наконец, отметим, что ценность книги как учебного пособия определяется и включением в нее двух первых приложений, довольно обстоятельно излагающих основы теории чисел и конечных полей и делающих книгу более независимой.

Общий абрис содержания книги дан в предисловии автора. Более детальную информацию о содержании можно почерпнуть в оглавлении. Мы коснемся лишь некоторых аспектов. До середины 1970-х годов эволюция криптографии, в основном, шла в рамках парадигмы транспозиционно-подстановочных шифров. Традиционные криптосистемы, реализующие такие шифры, называются также (симметричными) криптосистемами с единым ключом, поскольку в них для шифрования и дешифрования используется один и тот же секретный ключ. В таких системах распределение и обновление ключей трудны и создают угрозу безопасности.

Год 1976-й стал, так сказать, "точкой бифуркации" (или "ветвления") процесса развития криптографии: Диффи и Хеллман [DifH76] предложили общую революционную концепцию (асимметричных) криптосистем с публичными ключами (public key cryptosystems), а также дали пример подобной системы для выработки общего секретного ключа; соответствующий протокол использует публичные каналы связи (скажем, газетные объявления). В криптосистемах с публичными ключами каждый пользователь U сети имеет два ключа: секретный ключ D для дешифрования и создания цифровой подписи и публичный ключ E, с помощью которого другие пользователи шифруют сообщения для U и проверяют его подпись. Ключ E *публикуется*, например, в общедоступном каталоге на некотором сервере. (В России ключ E часто называют "открытым", что вызывает смысловой диссонанс с "открытым" текстом, который в системах секретности (т. е. системах шифрования-дешифрования) обычно — в отличие от публичного ключа — *секретен* либо защищен цифровой подписью или кодом аутентификации.) Начиная с конца 1970-х годов хлынул поток работ о криптосистемах с публичными ключами. Важно отметить, что безопасность всех таких систем зиждется на математических гипотезах о практической невозможности решения тех или иных вычислительных проблем. Например, система Диффи–Хеллмана основана на проблеме дискретного логарифмирования в конечном поле, RSA-система [RivSA78] — на проблеме факторизации составного числа, были разработаны и реализованы криптосистемы, основанные на кодах Гоп-

пы, исправляющих ошибки [McEl78], на NP-полной проблеме "укладки рюкзака" [MerH78] и т. д. Теперь в сферу криптологии оказались вовлечены теория чисел и алгебра, и это резкое расширение математиеского инструментария вызвало большой интерес не только у специалистов, но и у многих математиков, ранее далеких от криптологии. Заметим, что более половины объема основного текста книги ван Тилборга отведено криптосистемам с публичными ключами.

Главы 2, 4, 5, 6, 11, 12, 15 и приложение С перевел Д.С.Ананичев, а предисловие автора, главы 1, 3, 7, 8, 9, 10, 13, 14 и приложения A, B и D — И.О.Коряков. Окончательная компьютерная верстка в LaTeXe выполнена Д.С. Ананичевым (и здесь мы хотим поблагодарить А.А.Финогенова за ряд полезных советов). Список литературы был дополнен редактором рядом работ; они помечены звездочкой. Небольшие ошибки и опечатки исправлялись без какого-либо уведомления. Разумееется, некоторые ошибки могли сохраниться, а некоторые могли быть добавлены при переводе, за что мы, безусловно, несем полную ответственность. В ряде случаев возникала потребность сделать те или иные подстрочные примечания; они имеют сквозную нумерацию в каждой главе.

Надеемся, что эта книга окажется актуальной и найдет читателей среди математиков, инженеров и студентов, специализирующихся в области защиты информации, а также специалистов других профилей.

Замечания и пожелания по содержанию книги можно направлть по E-mail: Dmitry.Ananichev@usu.ru.

Екатеринбург, апрель 2004 *И.О.Коряков*

Предисловие

Защита секретной информации против неполномочного доступа и мошеннических изменений была первичной заботой на протяжении столетий. Современная техника коммуникации, использующая компьютеры, связанные через сети, делает все данные даже более уязвимыми для этих угроз. Возникли также новые вопросы, не стоявшие ранее, например, как добавить цифровую подпись к электронному документу таким образом, чтобы подписавший не мог впоследствии отрицать, что именно он подписал документ.

Криптология обращена к указанным выше вопросам. Она лежит в основе всей информационной безопасности. Техника, применяемая с этой целью, становится все более математической по своей природе. Эта книга служит введением в современные криптографические методы. После краткого обзора классических криптосистем мы сосредоточиваем внимание на трех главных областях. Прежде всего обсуждаются поточные и блочные шифры. Эти системы имеют крайне быстрые реализации, но отправитель и получатель должны разделять секретный ключ. Криптосистемы с публичными ключами (вторая главная область) делают возможной защиту данных без предсогласованного ключа. Их безопасность базируется на неподдающихся решению проблемах наподобие факторизации больших целых чисел. Остальные главы покрывают разнообразные темы, такие, как доказательства с нулевым разглашением, схемы разделения секрета и коды аутентификации. Два приложения довольно детально объясняют все предварительные математические сведения. Одно посвящено элементарной теории чисел (алгоритм Эвклида, китайская теорема об остатках, квадратичные вычеты, формулы обращения, непрерывные дроби). Другое приложение содержит введение в конечные поля и их алгебраическую структуру.

Эта книга отличается от ее версии 1988 г. в двух аспектах. То, что было добавлено много нового материала, следовало ожидать, поскольку данная область развивается очень быстро. Помимо ревизии имевшегося материала, включено много новых или сильно расширенных разделов, добавлены новые главы об эллиптических кривых и о кодах аутентификации. Второе отличие даже более существенно. Вся книга доступна в электронном виде как интерактивный учебник в пакете "Mathematica". Поэтому имеются сильные связи между различными местами текста, но более важно то, что теперь можно прорабатывать нетривиальные примеры. Даже не-эксперт легко может изменить параметры в примерах и испытать новые параметры. Наш опыт, основанный на преподавании в Калифорнийском технологическом институте и в Эйндховенском тех-

нологическом университете, показывает, что большинство студентов искренне наслаждаются огромными возможностями компьютерной алгебры. Всюду в этой книге нашим намерением было сделать предложения пакета "Mathematica" настолько прозрачными, насколько это возможно, иногда жертвуя элегантными или более разумными альтернативами, которые слишком зависят от этого конкретного пакета компьютерной алгебры.

Несколько человек сыграли решающую роль в подготовке этой книги. Перечисляя в алфавитном порядке (первых) имен, я хотел бы поблагодарить Fred'a Simons'a за демонстрацию мне полного потенциала пакета "Mathematica" для образовательных целей и за усиление многих команд пакета, Gavin'a Horn'a за многие найденные им опечатки, равно как и за составление решений, Lilian Porter за ее реакцию на мой английский, Wil'a Kortsmit'a за его помощь в получении готовой к печати рукописи и за разрешение многих моих вопросов о пакете "Mathematica". Я также многим обязан следующим людям, которые помогли мне своими откликами на различные главы: это Berry Schoenmakers, Bram van Asch, Eric Verheul, Frans Willems, Mariska Sas и Martin van Dijk.

Henk van Tilborg
Dept. of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven
the Netherlands
email: henkvt@win.tue.nl

Глава 1
Введение

1.1 Введение и терминология

Криптологию, науку о криптосистемах, можно подразделить на две дисциплины. *Криптография* занимается разработкой криптосистем, тогда как *криптоанализ* изучает способы взлома криптосистем. Эти два аспекта тесно связаны; при внедрении криптосистемы важную роль играет анализ ее безопасности. Пока мы не даем формального определения криптосистемы, оно появится в этой главе позже. Предполагается, что у читателя есть правильное интуитивное понимание того, что такое криптосистема.

Для чего нужны криптосистемы? Вот некоторые их возможности:

Конфиденциальность: при передаче данных нежелательно, чтобы какой-нибудь перехватчик понял содержание передаваемых сообщений. То же справедливо для сохраняемых данных, которые должны быть защищены от несанкционированного доступа, например, хакеров.

Аутентификация: это свойство эквивалентно цифровой подписи. Получатель сообщения хочет убедиться, что сообщение пришло от определенной стороны, а не от кого-либо иного (даже если позже первоначальная сторона захочет отрицать это).

Целостность: это означает, что получатель некоторых данных имеет свидетельство, что третьей стороной не было сделано никаких изменений.

На продолжении столетий (см. [Kahn67]) криптосистемы использовались военными и дипломатическими службами. В наши дни широкое распространение в промышленности сетей связи, управляемых компьютерами, использование последних в гражданских службах часто требует специальной защиты данных посредством криптографической техники.

Поскольку запись данных и последующее их извлечение можно рассматривать как передачу этих данных во врéменную область, мы всегда будем использовать термин “передача”, когда данные записываются или передаются.

1.2 Шенноновское описание традиционной криптосистемы

В главах 2, 3 и 4 обсуждаются так называемые традиционные криптосистемы. Формальное определение традиционной криптосистемы, так же как и математические основы соответствующей теории, принадлежат Шеннону [Shan49] (см. также *[Ящен98]). На рис. 1.1 изображена общая схема традиционной криптосистемы[1].

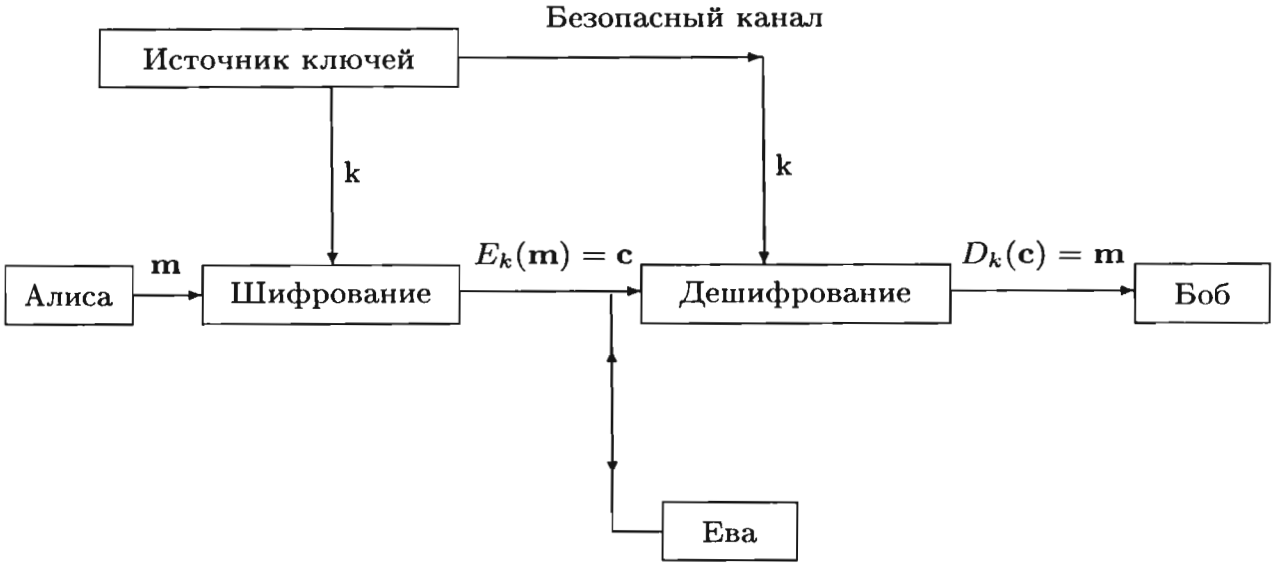


Рис. 1.1. Традиционная криптосистема.

Ниже мы детально остановимся на таких понятиях как язык и текст. Это обеспечит криптоаналитика полезными моделями для описания отправляемого сообщения в нашей схеме.

Пусть $\mathcal{A}$ — конечное множество, называемое *алфавитом*. Через $|\mathcal{A}|$ обозначим мощность $\mathcal{A}$. В качестве алфавита мы часто будем использовать множество $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$, работая с его элементами по модулю q (см. подраздел А.3.1 и раздел В.2). Алфавит $\mathbb{Z}_{26}$ можно отождествить с множеством $\{a, b, \ldots, z\}$. В большинстве современных приложений q равно 2 или степени двойки.

Конкатенация n букв из $\mathcal{A}$ будет называться *n-граммой* и обозначаться посредством $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})$. Частными случаями являются *биграммы* ($n = 2$) и *триграммы* ($n = 3$). Множество всех n-грамм над $\mathcal{A}$ обозначается через $\mathcal{A}^n$.

Текст — это элемент из $\mathcal{A}^* = \cup_{n \geq 0} \mathcal{A}^n$. *Язык* — это подмножество в $\mathcal{A}^*$. В случае языков программирования такие подмножества точно

[1] Некоторые российские криптографы (см., например, *[АЗКЧ01]) используют термин "дешифрование" для действия криптоаналитика, а соответствующее действие легитимного получателя называют "расшифрованием". При этом "шифрование" называют "зашифрованием". — *Прим. ред.*

определяются рекурсивными правилами. В случае разговорных языков правила весьма вольные.

Пусть $\mathcal{A}$ и $\mathcal{B}$ — конечные алфавиты. Любое инъективное (т.е. взаимно однозначное) отображение E из $\mathcal{A}^*$ в $\mathcal{B}^*$ называется *криптографическим преобразованием*. В большинстве практических ситуаций мощности $|\mathcal{A}|$ и $|\mathcal{B}|$ равны. Также часто криптографическое преобразование E отображает n-граммы в n-граммы (чтобы избежать расширения данных в процессе шифрования).

Пусть $\mathbf{m}$ — сообщение (текст из $\mathcal{A}^*$), которое Алиса на рис. 1.1 хочет секретно передать Бобу. Обычно оно называется *открытым текстом* (plaintext). Сначала Алиса преобразует открытый текст в так называемый *шифртекст* (ciphertext) (или *криптограмму*) $\mathbf{c} = E(\mathbf{m})$. Этот шифртекст она и передает Бобу.

Определение 1.1
Симметричная (или *традиционная*) *криптосистема* — это множество $\mathcal{E} = \{E_k | k \in \mathcal{K}\}$ криптографических преобразований. Индексирующее множество $\mathcal{K}$ называется *ключевым пространством*, а его элементы — *ключами*.

Так как отображение E_k инъективно, существует обратное к нему отображение. Обозначим его через D_k. Разумеется, E означает *шифрование* (encryption или enciphering), а D — *дешифрование* (decryption или deciphering). Имеем

$$D_k(E_k(\mathbf{m})) = \mathbf{m} \quad \text{для всех открытых текстов } \mathbf{m} \in \mathcal{A}^* \quad \text{и всех } k \in \mathcal{K}.$$

Если Алиса хочет послать Бобу открытый текст $\mathbf{m}$ с помощью криптографического преобразования E_k, то как Алиса, так и Боб должны знать конкретный выбор ключа k. Они должны согласовать значение k посредством так называемого *безопасного канала*. Этот канал может быть курьером, но возможно также, что Алиса и Боб заблаговременно согласовали выбор k.

Боб может дешифровать $\mathbf{c}$, вычисляя

$$D_k(\mathbf{c}) = D_k(E_k(\mathbf{m})) = \mathbf{m}.$$

Обычно одна и та же криптосистема $\mathcal{E}$ используется длительное время и многими людьми; поэтому разумно предполагать, что это множество криптографических преобразований известно и криптоаналитику. Чтобы обеспечить безопасность данных, нужно часто менять ключ. Этот принцип еще в 19-м веке ясно сформулировал голландец Огюст Керкхофф (см. [Kahn67]).

Криптоаналитик (скажем, Ева), подсоединившийся к линии связи, может быть

- *пассивным* (перехватчиком): криптоаналитик пытается найти $\mathbf{m}$ (а еще лучше k), исходя из $\mathbf{c}$ (и любых знаний, которыми он обладает). Определив k, можно вскрыть последующие шифртексты.

- *активным* (злоумышленником): криптоаналитик пытается активно манипулировать передаваемыми данными. Например, он передает собственный шифртекст, вновь передает старый шифртекст, подставляет свои собственные тексты вместо передаваемых шифртекстов и т.п.

В целом различают три уровня криптоанализа:

- *Атака только с шифртекстом*: криптоаналитику известен только отрывок шифртекста (и часто контекст сообщения).

- *Атака с известным открытым текстом*: известен отрывок шифртекста и соответствующий открытый текст. Если система безопасна относительно атак такого рода, легитимный получатель не обязан уничтожать дешифрованные сообщения.

- *Атака с выбранным открытым текстом*: криптоаналитик может выбрать любой открытый текст и сгенерировать соответствующий шифртекст. Криптосистемы, обсуждаемые в гл. 7–12, должны противостоять атакам такого рода.

Этим завершается общее описание традиционной криптосистемы, изображенной на рис. 1.1.

1.3 Статистическое описание источника открытых текстов

Как мы увидим в разд. 2.2, в криптологии, особенно когда хотят взломать конкретную криптосистему, мощным инструментом часто служит вероятностный подход к описанию языка.

Алиса на рис. 1.1 играет роль конечного или бесконечного *источника S открытых текстов* над алфавитом $\mathcal{A}$, например, над $\mathbb{Z}_q$. Источник можно описать как конечную (соответственно бесконечную) последовательность случайных переменных M_i, т.е. как последовательность

$$M_0, M_1, \ldots, M_{n-1} \quad \text{для некоторого фиксированного значения } n$$

(соответственно как

$$M_0, M_1, M_2, \ldots),$$

причем для всех появляющихся событий заданы вероятности. Так, для каждой комбинации букв (r-граммы) $(m_0, m_1, \ldots, m_{r-1})$ над $\mathcal{A}$ и каждой начальной точки j определена вероятность

$$\Pr_{\text{откр}}(M_j = m_0, M_{j+1} = m_1, \ldots, M_{j+r-1} = m_{r-1}).$$

Когда $j = 0$, мы пишем просто $\Pr_{\text{откр}}(m_0, m_1, \ldots, m_{r-1})$. Конечно, вероятности, описывающие источник открытых текстов, должны обладать некоторыми стандартными статистическими свойствами, которые мы упомянем, но не будем обсуждать в деталях:

1. $\Pr_{\text{откр}}(m_0, m_1, ..., m_{r-1}) \geq 0$ для всех текстов $(m_0, m_1, ..., m_{r-1})$;
2. $\sum_{(m_0, m_1, ..., m_{r-1})} \Pr_{\text{откр}}(m_0, m_1, ..., m_{r-1}) = 1$;
3. $\sum_{(m_r, m_{r+1}, ..., m_{l-1})} \Pr_{\text{откр}}(m_0, m_1, ..., m_{l-1}) = \Pr_{\text{откр}}(m_0, m_1, ..., m_{r-1})$ для всех $l > r$.

Третье свойство называется *условием совместности Колмогорова.*

Пример 1.1

Источник $\mathcal{S}$ открытых текстов (Алиса на рис. 1.1) генерирует отдельные буквы (1-граммы) из $\{a, b, \ldots, z\}$ с независимыми, но идентичными [по отношению к моментам появления — *ред.*] *распределениями, скажем, $p(a), p(b), \ldots, p(z)$. Таким образом,*

$$\Pr_{\text{откр}}(m_0, m_1, \ldots, m_{n-1}) = p(m_0)p(m_1), \ldots, p(m_{n-1}), \ n \geq 1.$$

Распределение букв алфавита в обычных английских текстах приведено в табл. 1.1 (см. табл. 12-1 в [MeyM82]*). В этой модели*

$$\Pr_{\text{откр}}(run) = p(r)p(u)p(n) = 0{,}612 \times 0{,}0271 \times 0{,}0709 \approx 1{,}18 \cdot 10^{-4}.$$

Заметим, что в этой модели также $\Pr_{\text{откр}}(nru) = p(n)p(r)p(u)$ и т.д, т.е. все перестановки трех букв r, u и n равновероятны, что неправдоподобно для обычных английских текстов.

A	0.0804	H	0.0549	O	0.0760	V	0.0099
B	0.0154	I	0.0726	P	0.0200	W	0.0192
C	0.0306	J	0.0016	Q	0.0011	X	0.0019
D	0.0399	K	0.0067	R	0.0612	Y	0.0173
E	0.1251	L	0.0414	S	0.0654	Z	0.0009
F	0.0230	M	0.0253	T	0.0925		
G	0.0196	N	0.0709	U	0.0271		

Таблица 1.1. Распределение вероятностей 1-грамм в английском языке.

Пример 1.2

$\mathcal{S}$ порождает 2-граммы над алфавитом $a, b, \ldots, z$ с независимыми, но идентичными распределениями, скажем, $p(s, t)$, где $s, t \in \{a, b, \ldots, z\}$. Таким образом, для $n \geq 1$

$$\Pr_{\text{откр}}(m_0, m_1, \ldots, m_{2n-1}) = p(m_0, m_1)p(m_2, m_3) \ldots p(m_{2n-2}, m_{2n-1}).$$

Распределение 2*-грамм в английских текстах можно найти в литературе (см. табл.* 2.3.4 *в* [Konh81]*).*

Конечно, можно продолжать подобным же образом с таблицами распределения 3-грамм и т.д. Иной, более привлекательный подход дается в следующем примере.

```
ed["a"] = 0.0723; ed["j"] = 0.0006; ed["s"] = 0.0715;
ed["b"] = 0.0060; ed["k"] = 0.0064; ed["t"] = 0.0773;
ed["c"] = 0.0282; ed["l"] = 0.0390; ed["u"] = 0.0272;
ed["d"] = 0.0483; ed["m"] = 0.0230; ed["v"] = 0.0117;
ed["e"] = 0.1566; ed["n"] = 0.0814; ed["w"] = 0.0078;
ed["f"] = 0.0167; ed["o"] = 0.0716; ed["x"] = 0.0030;
ed["g"] = 0.0216; ed["p"] = 0.0160; ed["y"] = 0.0168;
ed["h"] = 0.0420; ed["q"] = 0.0007; ed["z"] = 0.0010;
ed["i"] = 0.0787; ed["r"] = 0.0750;
```

Таблица 1.2. Равновесное распределение в английском языке.

Пример 1.3

В этой модели источник $\mathcal{S}$ *открытых текстов генерирует* 1*-граммы посредством марковского процесса* [точнее, марковской цепи с конечным числом состояний — *ред.*]. *Этот процесс можно описать матрицей переходов* $P = (p(t,s))_{s,t}$, *где* $p(t,s)$ — *вероятность того, что вслед за буквой* s *в тексте следует буква* t. *Из теории марковских процессов вытекает, что* P *имеет собственное значение* 1. *Пусть* $\mathbf{p} = (p(a), p(b), \ldots, p(z))$ — *соответствующий собственный вектор, называемый равновесным* [или стационарным, инвариантным — *ред.*] *распределением процесса. Предполагая, что процесс с самого начала находится в равновесном состоянии, получаем*

$$\Pr_{\text{откр}}(m_0, m_1, \ldots, m_{n-1}) = p(m_0)p(m_1|m_0)p(m_2|m_1)\ldots p(m_{n-1}|m_{n-2}).$$

Пусть $\mathbf{p}$ *и* P *заданы таблицами* 1.2 *и* 1.3 *(из* [Konh81]; *здесь они обозначены посредством "ed" и "TrPr"). Тогда получаются следующие более правдоподобные вероятности:*

$$\Pr_{\text{откр}}(run) = p(r)p(u|r)p(n|u) = 0.0751 \times 0.0192 \times 0.1517 \approx 2.18 \cdot 10^{-4},$$

$$\Pr_{\text{откр}}(urn) = p(u)p(r|u)p(n|r) = 0.0272 \times 0.1460 \times 0.0325 \approx 1.29 \cdot 10^{-4},$$

$$\Pr_{\text{откр}}(nru) = p(n)p(r|n)p(u|r) = 0.0814 \times 0.0011 \times 0.0192 \approx 1.72 \cdot 10^{-6}.$$

С помощью функций StringTake, ToCharacterCode *и* StringLength *пакета "Mathematica" эти вероятности могут быть вычислены следующим способом (сначала вводятся таблицы* 1.2 *и* 1.3*):*

```
sourcetext = "run";
ed[StringTake[sourcetext, {1}]] *
StringLength[sourcetext] - 1
        ∏                    TrPr[[
        i=1
    ToCharacterCode[StringTake[sourcetext, {i}]] - 96,
    ToCharacterCode[StringTake[sourcetext, {i + 1}]] - 96]]
```

```
|| {{0.000218448}}
```

Можно получить лучшие аппроксимации языка, если считать вероятности переходов зависящими от более чем одной предшествующей буквы.

Отметим, что во всех трех рассмотренных примерах модели *стационарны*, т.е. $\Pr_{\text{откр}}(M_j = m_0, M_{j+1} = m_1, \ldots, M_{j+n-1} = m_{n-1})$ не зависит от значения j. Можно считать, что в середине обычного текста это свойство выполняется, но в иных ситуациях это не так. Подумайте, например, о дате в начале письма.

	a	b	c	d	e	f	g	h	i
a	0.0011	0.0193	0.0388	0.0469	0.002	0.01	0.0233	0.002	0.048
b	0.0931	0.0057	0.0016	0.0008	0.3219	0	0	0	0.0605
c	0.1202	0	0.0196	0.0004	0.1707	0	0	0.1277	0.0761
d	0.1044	0.002	0.0026	0.0218	0.3778	0.0007	0.0132	0.0007	0.1803
e	0.066	0.0036	0.0433	0.1194	0.0438	0.0142	0.0125	0.0021	0.0158
f	0.0838	0	0	0	0.1283	0.0924	0	0	0.1608
g	0.1078	0	0	0.0018	0.2394	0	0.0177	0.1281	0.0839
h	0.1769	0.0005	0.0014	0.0008	0.5623	0	0	0.0005	0.1167
i	0.038	0.0082	0.0767	0.0459	0.0437	0.0129	0.028	0.0002	0.0016
j	0.1259	0	0	0	0.1818	0	0	0	0.035
k	0.0395	0.0028	0	0.0028	0.5282	0.0028	0	0.0198	0.1582
l	0.1342	0.0019	0.0022	0.0736	0.1918	0.0105	0.0108	0	0.1521
m	0.1822	0.0337	0.0026	0	0.2975	0.001	0	0	0.1345
n	0.055	0.0004	0.0621	0.1681	0.1212	0.0102	0.1391	0.0013	0.0665
o	0.0085	0.0101	0.0162	0.0231	0.0037	0.1299	0.0082	0.0025	0.0092
p	0.1359	0	0.0006	0	0.1747	0	0	0.0237	0.0423
q	0	0	0	0	0	0	0	0	0
r	0.1026	0.0033	0.0172	0.0282	0.2795	0.0031	0.0175	0.0017	0.1181
s	0.0604	0.0012	0.0284	0.0027	0.1795	0.0024	0	0.0561	0.1177
t	0.0619	0.0003	0.0036	0.0002	0.1417	0.0007	0.0002	0.3512	0.1406
u	0.0344	0.0415	0.0491	0.0243	0.0434	0.0052	0.0382	0.001	0.0258
v	0.0749	0	0	0.0023	0.6014	0	0	0	0.2569
w	0.2291	0.0008	0	0.0032	0.1942	0	0	0.1422	0.2104

Таблица 1.3 (начало). Вероятности переходов $p(t|s)$, s — строка, t — столбец, в английском языке.

	a	b	c	d	e	f	g	h	i
x	0.0672	0	0.1119	0	0.1269	0	0	0.0075	0.1119
y	0.0586	0.0034	0.0103	0.0069	0.2897	0	0	0	0.069
z	0.2278	0	0	0	0.4557	0	0	0	0.2152

	j	k	l	m	n	o	p	q	r
a	0.002	0.0103	0.1052	0.0281	0.1878	0.0008	0.0222	0	0.118
b	0.0057	0	0.1242	0.0049	0	0.0964	0	0	0.0662
c	0	0.0324	0.0369	0.0015	0.0011	0.2283	0	0.0004	0.0426
d	0.0033	0	0.0125	0.0178	0.0053	0.0733	0	0.0007	0.0324
e	0.0005	0.0036	0.0456	0.034	0.1381	0.004	0.0192	0.0034	0.1927
f	0	0	0.0299	0.0009	0.0009	0.2789	0	0	0.1215
g	0	0	0.0203	0.0027	0.0451	0.114	0	0	0.1325
h	0	0	0.0016	0.0016	0.0038	0.0786	0	0	0.0153
i	0	0.005	0.0567	0.0297	0.2498	0.0893	0.01	0.0008	0.0342
j	0	0	0	0	0	0.3147	0	0	0.007
k	0	0.0113	0.0198	0.0028	0.0565	0.0198	0	0	0.0085
l	0	0.0079	0.1413	0.0082	0.0004	0.0778	0.0041	0	0.0034
m	0	0	0.001	0.0654	0.0042	0.1246	0.0722	0	0.0026
n	0.0009	0.0066	0.0073	0.0104	0.0194	0.0528	0.0004	0.0007	0.0011
o	0.0014	0.0078	0.0416	0.0706	0.219	0.0222	0.0292	0	0.153
p	0	0	0.0812	0.0073	0.0006	0.1511	0.0581	0	0.2306
q	0	0	0	0	0	0	0	0	0
r	0	0.0205	0.0164	0.0303	0.0325	0.1114	0.0055	0	0.0212
s	0	0.0091	0.0145	0.0112	0.0021	0.0706	0.0386	0.0009	0.0027
t	0	0	0.0101	0.0044	0.0015	0.1229	0.0003	0	0.0479
u	0	0.0014	0.1097	0.0329	0.1517	0.0019	0.0386	0	0.146
v	0	0	0	0.0012	0	0.053	0	0	0
w	0	0	0.0041	0	0.0357	0.1292	0	0	0.0106
x	0	0	0	0.0075	0	0.0075	0.3507	0	0
y	0	0.0034	0.0172	0.0379	0.0172	0.2207	0.031	0	0.031
z	0	0	0.127	0	0	0.0506	0	0	0

	s	t	u	v	w	x	y	z
a	0.1001	0.1574	0.0137	0.0212	.0057	0.0026	0.0312	0.0023
b	0.0229	0.0049	0.0727	0.0016	0	0	0.1168	0
c	0.0087	0.0893	0.0347	0	0	0	0.0094	0
d	0.0495	0.0013	0.0601	0.0099	0.004	0	0.0264	0
e	0.1231	0.0404	0.0048	0.0215	0.0205	0.0152	0.0121	0.0004
f	0.0026	0.0496	0.0462	0	0	0	0.0043	0
g	0.0256	0.0247	0.0512	0	0	0	0.0053	0
h	0.0027	0.0233	0.0085	0	0.0011	0	0.0041	0
i	0.1194	0.1135	0.0011	0.025	0	0.0023	0.0002	0.0079

Таблица 1.3 (продолжение). Вероятности переходов $p(t|s)$, s — строка, t — столбец, в английском языке.

	s	t	u	v	w	x	y	z
j	0	0	0.3357	0	0	0	0	0
k	0.1102	0.0028	0.0028	0	0	0	0.0113	0
l	0.0389	0.0254	0.0269	0.0056	0.0011	0	0.0819	0
m	0.0244	0.0005	0.0337	0.0005	0	0	0.0192	0
n	0.0751	0.1641	0.0124	0.0068	0.0018	0.0002	0.0157	0.0004
o	0.0357	0.0396	0.0947	0.0334	0.0345	0.0012	0.0041	0.0004
p	0.018	0.0287	0.0457	0	0	0	0.0017	0
q	0	0	1	0	0	0	0	0
r	0.0655	0.0596	0.0596	0.0192	0.0142	0.0017	0.0002	0.0306
s	0.0836	0.2483	0.0579	0	0.0039	0	0.0081	0
t	0.0418	0.0213	0.0195	0.0005	0.0088	0	0.0203	0.0005
u	0.1221	0.1255	0.0029	0.0014	0	0.001	0.0014	0.0005
v	0.0023	0	0.0012	0.0012	0	0	0.0058	0
w	0.0366	0.0016	0	0	0	0	0.0024	0
x	0	0.1716	0	0	0	0.0373	0	0
y	0.1517	0.0172	0.0138	0	0.0103	0	0.0069	0.0034
z	0	0	0.0127	0	0	0	0	0.0253

Таблица 1.3 (окончание). Вероятности переходов $p(t|s)$, s — строка, t — столбец, в английском языке.

1.4 Задачи

Задача 1.1. Какова вероятность получить текст “apple”, если источник открытых текстов генерирует независимые, одинаково распределенные 1-граммы, как указано в примере 1.1? Ответьте на тот же вопрос, когда используется марковская модель из примера 1.3.

Задача 1.2^M. Используйте функцию *Permutations* пакета “Mathematica” и формулу из примера 1.3, чтобы для каждого из 24 упорядочений четырех букв e, h, l, p определить вероятность того, что оно появится в языке, порожденном марковской моделью из примера 1.3.

Глава 2

Классические криптосистемы

2.1 Шифр Цезаря, простая замена, шифр Виженера

В этом разделе мы обсудим ряд классических криптосистем. Желающие расширить свой кругозор в этой области могут обратиться к [BekP82], [Denn82], [Kahn67], [Konh81] или [MeyM82].

2.1.1 Шифр Цезаря

Одна из старейших криптосистем обязана своим появлением Юлию Цезарю. В ней каждая буква текста сдвигается циклически на k позиций по алфавиту. Например, при $k = 7$ получается следующее шифрование слова "cleopatra" ("циклически" означает, что за буквой z следует a).

$$\text{cleopatra} \xrightarrow{+1} \text{dmfpqbusb} \xrightarrow{+1} \text{engqrcvtc} \xrightarrow{+1} \text{fohrsdwud} \xrightarrow{+1} \text{gpistexve}$$

$$\xrightarrow{+1} \text{hqjtufywf} \xrightarrow{+1} \text{irkuvgzxg} \xrightarrow{+1} \text{jslvwhayh}$$

Используя функции *ToCharacterCode* и *FromCharacterCode* пакета "Mathematica", преобразующие символы в их ASCII коды и обратно (код буквы a — 97, код буквы b — 98, и т.д.), можно задать шифр Цезаря с помощью следующей функции:

```
CaesarCipher[plaintext_, key_] :=
FromCharacterCode[
Mod[ToCharacterCode[plaintext] - 97 + key, 26] + 97]
```

Теперь — пример использования:

```
plaintext = "typehereyourplaintextinsmallletters";
key = 24;
CaesarCipher[plaintext, key]
```

```
rwncfcpcwmspnjyglrcvrglqkyjjjcrrcpq
```

В терминах раздела 1.2 *шифр Цезаря* определяется над алфавитом $\{0, 1, ..., 25\}$ равенствами:

$$E_k(m) = (m + k) \bmod 26, \ 0 \leq m < 26,$$

$$\mathcal{E} = \{E_k \mid 0 \leq k < 26\},$$

где через $i \bmod n$ обозначено единственное целое число j, $0 \leq j < n$, удовлетворяющее условию $j \equiv i \pmod{n}$. В этом случае пространством $\mathcal{K}$ является множество $\{0, 1, ..., 25\}$ и $D_k = E_{26-k}$.

Эту криптосистему легко взломать, просто перебрав все ключи. Такой метод называется *полным перебором ключей*. В таблице 2.1 можно найти пример криптоанализа шифровки "xyuysuyifvyxi".

x	y	u	y	s	u	y	i	f	v	y	x	i
w	x	t	x	r	t	x	h	e	u	x	w	h
v	w	s	w	q	s	w	g	d	t	w	v	g
u	v	r	v	p	r	v	f	c	s	v	u	f
t	u	q	u	o	q	u	e	b	r	u	t	e

Таблица 2.1. Криптоанализ шифра Цезаря.

Чтобы расшифровать криптограмму yhaklwpnw, можно перебрать все ключи в определенной выше функции Цезаря.

```
ciphertext = "yhaklwpnw"
Table[CaesarCipher[ciphertext, -key], {key, 1, 26}
```

```
{xgzjkvomv, wfyijunlu, vexhitmkt, udwghsljs, tcvfgrkir,
sbuefqjhq, ratdepigp, qzscdohfo, pyrbcngen, oxqabmfdm,
nwpzalecl, mvoyzkdbk, lunxyjcaj, ktmwxibzi, jslvwhayh,
irkuvgzxg, hqjtufywf, gpistexve, fohrsdwud, engqrcvtc,
dmfpqbusb, cleopatra, bkdnozsqz, ajcmnyrpy, ziblmxqox,
yhaklwpnw}
```

2.1.2 Шифр простой замены

□ Система и ее основное слабое место

Метод *простой замены* состоит в том, что фиксируется некоторая перестановка π алфавита $\{a, b, ..., z\}$, после чего она применяется к каждой букве открытого текста.

Пример 2.1

В следующем примере описана лишь часть перестановки π, достаточная для данного открытого текста. Используется функция StringReplace пакета "Mathematica".

```
StringReplace["plaintext",
{ "a" -> "k", "e" -> "z", "i" -> "b", "l" -> "r",
"n" -> "a", "p" -> "v", "t" -> "q", "x" -> "d" }]
```

‖ vrkbaqzdq

Более формальное описание системы простой замены состоит в следующем: пространством ключей $\mathcal{K}$ является множество S_q всех перестановок на множестве $\{0, 1, ..., q-1\}$, а криптосистема $\mathcal{E}$ определяется как

$$\mathcal{E} = \{E_\pi \mid \pi \in S_q\},$$

где

$$E_\pi(m) = \pi(m),\ 0 \leq m < q.$$

Функция дешифрования D_π задается равенством $D_\pi = E_{\pi^{-1}}$, поскольку

$$D_\pi(E_\pi(m)) = D(\pi(m)) = E_{\pi^{-1}}(\pi(m)) = \pi^{-1}(\pi(m)) = m,\ 0 \leq m < q.$$

В отличие от шифра Цезаря эта система свободна от такого недостатка, как слишком малое пространство ключей. Действительно, $|\mathcal{K}| = |S_{26}| = 26! \approx 4.03 \cdot 10^{26}$. Эта система дает великолепный пример того, как глупо верить в надежность криптосистемы, основываясь лишь на том, что она имеет большое пространство ключей! Простой подсчет частот появления букв в шифровке и их сравнение с частотами букв из табл. 1.1 позволяет быстро найти образы относительно π для большинства наиболее употребительных букв открытого текста. В самом деле, наиболее частая буква в криптограмме скорее всего окажется образом буквы e при перестановке π. Следующая по частоте будет образом буквы n и т.д.[1] После того как найдены прообразы наиболее частых букв криптограммы, нетрудно найти и остальные. Разумеется, чем длиннее криптограмма, тем легче ее расшифровать. В главе 5 мы вернемся к криптоанализу систем, в частности, к вопросу о том, как долго можно безопасно использовать один и тот же ключ.

□ Криптоанализ методом поиска наиболее вероятных слов

В следующем примере известен очень длинный шифртекст. Вообще говоря, для криптоанализа этого не требуется, но это необходимо, если мы хотим узнать весь ключ полностью. В самом деле, если длина шифртекста мала, то в нем может не встретиться какая-либо пара букв, и тем самым по нему невозможно полностью восстановить ключ, но, разумеется, для его расшифровки весь ключ и не нужен. Помимо известного шифртекста, приведенного в табл. 2.2, будем считать известным то, что в открытом тексте шла речь о "теории двусторонней связи" (по английски "bidirectional communication theory"). Тогда расшифровка еще больше упрощается.

[1] В русском языке наиболее частыми являются буквы о и е. — *Прим. перев.*

ndize	hicle	osiol	digic	imhzq	zolyi	zehdp	zhjeo	ndize
hlpvs	uczyc	dhzhj	eondi	zehde	moylk	zhjpm	lhylg	gidiz
ppsdo	lylzr	losye	nnmhz	ydise	hicle	osceu	lrloq	lgyoz
lneol	flhlo	dpydg	lzhuc	zyciu	eeone	olzhj	eondi	zehde
zhjpm	lhyll	dycei	clogi	dizgi	zydpp	siclq	zolyi	zehej
hjpml	hylzg	lkaol	gglqv	sqzol	yilqi	odhgj	eondi	zehxm
zlguc	zycyd	hehps	vlqlo	zrlqz	jiclp	duejy	dmgdp	ziszg
rlqqz	gizhf	mzqcz	hficl	ldopz	loydm	gljoe	niclp	dilol
zhvze	pefsd	hqgey	zepef	syenn	mhzyd	izehi	cleos	gllng
luzql	daapz	ydize	hgqml	ieicl	ydyii	cdipz	rzhfv	lzhfg
iclzo	dyize	hggem	oylge	jzhje	ondiz	ehucz	yczhj	pmlhy
eiclo	zhdpp	aeggz	vplqz	olyiz	ehgic	laolg	lhiad	aloql
gicly	dglej	vzqzo	lyize	hdpye	nnmhz	ydize	hicle	osdaa
eiclg	eyzdp	vlcdr	remoe	jneht	lsg...			

Таблица 2.2. Криптограмма, полученная простой заменой.

Предположив, что в открытом тексте встречается слово “communication”, ищем в шифртексте последовательность из 13 букв, в которой первая буква совпадает с восьмой, вторая — с двенадцатой, третья – с четвертой, шестая — с тринадцатой, а седьмая — с одиннадцатой.

И на самом деле, видим, что подходящая строка “yennmhzydizeh” трижды встречается в криптограмме. Отсюда извлекаем следующую информацию о перестановке π:

c	o	m	u	n	i	a	t
↓	↓	↓	↓	↓	↓	↓	↓
y	e	n	m	h	z	d	i

Предположив, что в открытом тексте встречается также слово “direction”, и используя уже полученную о π информацию, ищем в шифртексте последовательность вида “*z**yizeh”. Оказывается, что в шифровке четырежды встречается последовательность “qzolyizeh”, из чего следует, что в перестановке π

d	r	e
↓	↓	↓
q	o	l

Если теперь применить полученную информацию ко всему шифртексту, то легко найти перестановку π целиком. Например, начало текста имеет вид

in*ormationt*eor*treat*t*eunid...,

что, очевидно, читается как

information theory treats the unid(irectional)...

Это дает π-образы букв f, h и s. Продолжая подобным образом, легко получить всю перестановку:

a	b	c	d	e	f	g	h	i	j	k	l	m
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
d	v	y	q	l	j	f	c	z	w	t	p	n

n	o	p	q	r	s	t	u	v	w	x	y	z
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
h	e	a	x	o	g	i	m	r	u	k	s	b

Пример 2.2

Пакет "Mathematica" позволяет очень легко найти в данной строке подстроку с заданными свойствами. Например, для того, чтобы узнать, где в данной строке находится подстрока длины 6, *в которой* 1*-я буква совпадает с* 4*-й, а* 2*-я — с* 5*-й (как в латинском слове "quoque"), можно использовать функции If, StringTake, StringLength, Do и Print пакета "Mathematica" следующим образом:*

```
ciphertext = "xyuysuyifvyxi"
Do[
If[StringTake[ciphertext, {i + 1}] == StringTake[ciphertext,
{i + 4}]∧ StringTake[ciphertext, {i + 2}] ==
StringTake[ciphertext, {i + 5}],
Print[i + 1, " ", StringTake[ciphertext, {i + 1, i + 6}]]],
{i, 0, StringLength[ciphertext] - 6}]
```

```
|| uysuyi
```

Этот пример взят из табл. 2.1.

2.1.3 Криптосистема Виженера

Криптосистема Виженера (названная в честь Блеза де Виженера, который в 1586 г. в своем "Трактате о шифрах" описал более сложную версию подобной системы) состоит из r периодически применяемых шифров Цезаря. В приведенном ниже примере ключом является слово длины $r = 7$. В этом слове буква с номером i определяет частичный шифр Цезаря, т.е. используется для шифрования букв открытого текста с номерами i, $i + r$, $i + 2r$, ...

Пример 2.3

Отождествим $\{0, 1, ..., 25\}$ *с* $\{a, b, ..., z\}$. *Для шифрования и дешифрования весьма удобна так называемая таблица Виженера (см. табл.* 2.3*). Используя ключ "michael", выполняем следующее шифрование:*

открытый текст	a	c	r	y	p	t	o	s	y	s	t
ключ	m	i	c	h	a	e	l	m	i	c	h
шифртекст	m	k	t	f	p	x	z	e	g	u	a
открытый текст	e	m	o	f	t	e	n	i	s	a	c
ключ	a	e	l	m	i	c	h	a	e	l	m
шифртекст	e	q	z	r	b	g	u	i	w	l	o

0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
2	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
4	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
5	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
6	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
7	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
8	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
9	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
10	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
11	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
12	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
13	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
14	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
15	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
16	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
17	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
18	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
19	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
20	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
21	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
22	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
23	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
24	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
25	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Таблица 2.3. Таблица Виженера.

Из-за избыточности английского языка эффективный размер ключевого пространства существенно сократится, если в качестве ключей выбирать лишь осмысленные слова. Если же брать в качестве ключей имена родственников, как это было проделано в примере, то безопасность шифрования будет сведена практически к нулю.

Реализация в пакете "Mathematica" сложения двух букв, определяемого в таблице Виженера, аналогична приведенной выше реализации шифра Цезаря.

```
AddTwoLetters[a_, b_] :=
FromCharacterCode[Mod[(ToCharacterCode[a] - 97) +
(ToCharacterCode[b] - 97), 26] + 97]
```

Используя функции StringTake и StringLength пакета "Mathematica" и определенную выше функцию AddTwoLetters, можно реализовать шифрование в криптосистеме Виженера следующим образом:

```
plaintext = "typehereyourplaintextinsmallletters";
key = "keyword";
ciphertext = " ";
Do[ciphertext = ciphertext <>
AddTwoLetters[StringTake[plaintext, { i}],

StringTake[key,
{ Mod[i - 1, StringLength[key]] + 1} ]],
{ i, 1, StringLength[plaintext]} ];
ciphertext
```

```
dcnavvuocmqfgokmlpsowsrqiocovirpsiv
```

Более формальное описание криптосистемы Виженера дается следующим образом:

$$\mathcal{E} = \{E_{(k_0,k_1,\ldots,k_{r-1})} \mid (k_0, k_1, \ldots, k_{r-1}) \in \mathcal{K} = \mathbb{Z}_{26}^r\},$$

$$E_{(k_0,k_1,\ldots,k_{r-1})}(m_0, m_1, m_2, \ldots) = (c_0, c_1, c_2, \ldots),$$

где

$$c_i = (m_i + k_{(i \bmod r)}) \mod 26.$$

Вместо периодического использования r шифров Цезаря в криптосистеме Виженера можно, разумеется, применить и r произвольных простых замен. Такая система является примером так называемой *многоалфавитной замены*. Несколько веков не было эффективного способа взлома этой системы, в основном, из-за отсутствия техники определения длины ключа r. Ведь если удается определить r, то можно сгруппировать буквы $i, i+r, i+2r, \ldots$ для каждого i из интервала $0 \leq i < r$ и найти (например, частотным анализом) свою замену для каждой из этих групп по отдельности. В 1863 г. прусский офицер Фридрих В. Казиски указал статистический метод нахождения длины r ключа. В следующем разделе мы обсудим этот метод.

2.2 Подсчет числа совпадений, метод Казиски

2.2.1 Подсчет числа совпадений

Рассмотрим шифртекст $\mathbf{c} = c_0, c_1, \ldots, c_{n-1}$, полученный при шифровании шифром Виженера с ключом $\mathbf{k} = k_0, k_1, \ldots, k_{r-1}$ открытого текста $\mathbf{m} = m_0, m_1, \ldots, m_{n-1}$ на английском языке (см. также (2.1)). Как уже

было объяснено в конце предыдущего раздела, основой взлома криптосистемы Виженера является определение длины ключа r.

При проведении анализа будем предполагать, что источник открытых текстов генерирует отдельные буквы, причем вероятность появления каждой буквы задана в табл. 1.1 и не зависит от предшествующего текста (см. пример 1.1). Кроме того, предположим, что буквы k_i из ключа представляют собой независимые случайные величины, равномерно распределенные на множестве $\{a, b, \ldots, z\}$ (т.е. вероятность того, что k_i окажется некоторой наперед заданной буквой, равна 1/26).

Обозначим через $\mathbf{c}_{left}^{(i)}$ и $\mathbf{c}_{right}^{(i)}$ подстроки строки $\mathbf{c}$, состоящие из i крайних левых символов $\mathbf{c}$ и, соответственно, из i крайних правых символов $\mathbf{c}$, т.е.

$$\mathbf{c}_{left}^{(i)} = c_0, c_1, ..., c_{i-1} \quad \text{и} \quad \mathbf{c}_{right}^{(i)} = c_{n-i}, c_{n-i+1}, ..., c_{n-1}.$$

Затем подсчитаем количество совпадений символов строк $\mathbf{c}_{left}^{(i)}$ и $\mathbf{c}_{right}^{(i)}$, т.е. таких координат j, что $(\mathbf{c}_{left}^{(i)})_j = (\mathbf{c}_{right}^{(i)})_j$. В лемме 2.1 будет показано, что математическое ожидание доли этих совпадений (т.е. их количества, деленного на i, — длину строки) равно 0.06875, если r делит $n - i$, и равно $1/26 = 0.03846$, если r не делит $n - i$.

Приведем пример того, как можно использовать эту разницу в математических ожиданиях для определения длины r неизвестного ключа.

Пример 2.4

Рассмотрим криптограмму
"ubsyvkmhvyrrtsbbcrdsndwrtshxmbufrmxgabnvmircewerucam
lyzbrvfwivvmlyzwapsryogsslechbgcubsvyczqrcwrmhvcxgooyvcy
dspomtqfpyqkgbcmerucadlcaflrsuqjrbhceqesfcehuoqmdstorcdoy
meqqwaglgovggsmdabbigztbbqyfwbxwmgfpowgztyeilosrkgfahuo
vqfogswruqnvpwfvrnmpqqgsslatgrmqubsvyczqrswcjdeowqqroih
qdspdibffnxwgztbbqyfwbxus" ;

С помощью функций StringTake, StringLength, Characters и Table пакета "Mathematica" легко вычислить количество совпадений символов строк $\mathbf{c}_{left}^{(i)}$ и $\mathbf{c}_{right}^{(i)}$ для каждого значения i:

```
ciphertext =
"ubsyvkmhvyrrtsbbcrdsndwrtshxmbufrmxgabnvmirceweruca
mlyzbrvfwivvmlyzwapsryogsslechbgcubsvyczqrcwrmhvcxgo
oyvcydspomtqfpyqkgbcmerucadlcaflrsuqjrbhceqesfcehuoq
mdstorcdoymeqqwaglgovggsmdabbigztbbqyfwbxwmgfpowgzty
eilosrkgfahuovqfogswruqnvpwfvrnmpqqgsslatgrmqubsvycz
qrswcjdeowqqroihqdspdibffnxwgztbbqyfwbxus" ;
```

```
L = StringLength[ciphertext];
Table[ N[ Count[ Characters[ StringTake[ciphertext, i]]
- Characters[ StringTake[ciphertext, -i]], 0]/i,
1], {i, L - 20, L - 1}]
```

```
{0.03, 0.04, 0.08, 0.02, 0.05, 0.04, 0.04, 0.03, 0.06,
0.07, 0.06, 0.06, 0.04, 0.02, 0.05, 0.08, 0.04, 0.05,
0.02, 0.01, 0.05}
```

Наибольшие величины оказались в позициях этого списка с номерами -6 *и* -18*, а это показывает, что длина ключа* r *равна* 6*. И на самом деле, при шифровании был использован шестибуквенный ключ "monkey".*

Это можно проверить с помощью следующего аналога шифрования Виженера из примера 2.3 :

```
SubTwoLetters[a_, b_] :=
FromCharacterCode[
Mod[(ToCharacterCode[a] - 97) - (ToCharacterCode[b] -
97), 26] + 97]

ciphertext =
"ubsyvkmhvyrrtsbbcrdsndwrtshxmbufrmxgabnvmircewerucamly
zbrvfwivvmlyzwapsryogsslechbgcubsvyczqrcwrmhvcxgooyv
cydspomtqfpyqkgbcmerucadlcaflrsuqjrbhceqesfcehuoqmds
torcdoymeqqwaglgovggsmdabbigztbbqyfwbxwmgfpowgztyeil
osrkgfahuovqfogswruqnvpwfvrnmpqqgsslatgrmqubsvyczqrs
wcjdeowqqroihqdspdibffnxwgztbbqyfwbxus" ;
key = "monkey";
plaintext = "";
Do[plaintext = plaintext <>
SubTwoLetters[StringTake[cipertext, { i}],

StringTake[key,
{ Mod[i - 1, StringLength[key]] + 1} ]],
{ i, 1, StringLength[ciphertext]} ];
plaintext
```

```
informationtheorytreatstheunidirectionalinformation
channelbywhichaninformationsourceinfluencesstatisti
callyareceivercommunpcationtheoryhoweverdescribesth
emoregeneralcaseinwhichtwoormoreinformationsourcesi
nfluenceeachotherstatisticallythedirectionofthisinf
luenceisexpressedbydsrectedtransinformationqu
```

Лемма 2.1
Пусть шифртекст $\mathbf{c}$ получен при шифровании открытого текста $\mathbf{m}$ шифром Виженера с ключом $\mathbf{k}$ длины r. Пусть текст $\mathbf{m}$ порожден источником открытых текстов из примера 1.1. Иначе

говоря, буквы в $\mathbf{m}$ представляют собой независимые случайные величины, распределение вероятностей $p(m)$ которых определяется в соответствии с табл. 1.1. Кроме того, пусть буквы k_i из ключа — это независимые случайные величины, равномерно распределенные на множестве $\{a, b, \ldots, z\}$ (т.е. вероятность того, что k_i окажется некоторой наперед заданной буквой, равна $1/26$). Тогда для каждой пары (i, j), $1 \leq i < j \leq n$,

$$\Pr[c_i = c_j] = \begin{cases} \sum_m p(m)^2 \approx 0.06875, & \text{если } r \text{ делит } j - i, \\ 1/26 \approx 0.03846, & \text{если } r \text{ не делит } j - i. \end{cases}$$

Доказательство. Если $j - i$ делится на r, то $c_i = c_j$ тогда и только тогда, когда $m_i = m_j$. Это непосредственно следует из формулы (2.1), поскольку $j \bmod r$ равно $i \bmod r$. Таким образом,

$$\Pr[c_i = c_j] = \Pr[m_i = m_j] = \sum_m \Pr[m_i = m_j = m] =$$

$$= \sum_m \Pr[m_i = m] \cdot \Pr[m_j = m] = \sum_m p(m)^2 \approx 0.06875.$$

Если $j - i$ не делится на r, то по формуле (2.1) $c_i = c_j$ тогда и только тогда, когда $m_i + k_{i \bmod r} = m_j + k_{j \bmod r}$. Из неравенства $j \bmod r \neq i \bmod r$ следует, что $k_{j \bmod r}$ совпадает с $m_i + k_{i \bmod r} - m_j$ с вероятностью $1/26$. Отсюда

$$\Pr[c_i = c_j] = 1/26 \approx 0.03846.$$

■

Понятно, что чем больше длина криптограммы, тем легче определить длину ключа по долям совпадений символов в парах строк $\mathbf{c}_{left}^{(i)}$, $\mathbf{c}_{right}^{(i)}$ для различных i.

2.2.2 Метод Казиски

Криптоанализ Казиски криптосистемы Виженера основан на том факте, что если некоторая комбинация букв (часто встречающийся фрагмент открытого текста) шифруется более одного раза одним и тем же фрагментом ключа (это происходит из-за повторяемости ключа длины r), то в криптограмме на соответствующих местах возникают повторяющиеся куски.

Процитируем пример из [Baue97]:

Пример 2.5

Рассмотрим следующий открытый текст и криптограмму, полученную из него с помощью ключа "comet".

открытый текст	t	h	e	r	e	i	s	a	n	o	t
ключ	c	o	m	e	t	c	o	m	e	t	c
шифртекст	v	v	q	v	x	k	g	m	r	h	v
открытый текст	h	e	r	f	a	m	o	u	s	p	i
ключ	o	m	e	t	c	o	m	e	t	c	o
шифртекст	v	q	v	y	c	a	a	y	l	r	w

В криптограмме имеется два вхождения подстроки "vvqv", начинающиеся с 1*-й и с* 11*-й позиций. Это показывает, что длина ключа* r *делит* 10*. Кроме того, дважды встречается и подстрока "mrh", с* 8*-й и с* 23*-й позиций. Таким образом, кажется правдоподобным, что длина ключа* r *делит* 15*. Комбинируя эти результаты, можно сделать вывод, что* $r = 5$*. В данном случае вывод оказывается верен.*

Дополнительные подробности криптоанализа системы Виженера можно найти в [Baue97].

2.3 Шифры Вернама и Плэйфэра, перестановки, машина Хагелина, "Энигма"

В этом разделе мы кратко обсудим еще несколько криптосистем, не слишком углубляясь в их структуры.

2.3.1 Одноразовый щит

Одноразовый щит, иногда также называемый *шифром Вернама* (по имени Г.С. Вернама — служащего американской компании A.T.& T., который ввел эту систему в 1917г.), представляет собой шифр Виженера, в котором длина ключа равна длине открытого текста. При этом ключевая последовательность должна строится совершенно случайным образом и использоваться только один раз. Интуитивно ясно, что получающаяся в таком случае система *безусловно безопасна* (или *абсолютно стойка*), что будет доказано в гл. 5. Эта криптосистема применяется в "горячей линии" между Москвой и Вашингтоном. Главным недостатком такой системы является большая длина ключа, из-за чего система оказывается неудобной в большинстве приложений.

2.3.2 Шифр Плэйфэра

Шифр Плэйфэра (1854г., назван по имени шотландца Л.Плэйфэра) использовался Британией во время Первой мировой войны. Он работает с биграммами. Сперва отождествляются буквы i и j. Получившиеся 25 букв алфавита расставляются по строкам матрицы K размером 5×5 следующим образом. Ключевое слово выписывается по строкам, начиная с левого верхнего угла. В ключевое слово каждая буква должна входить не более одного раза. В противном случае повторы придется опустить. Не

вошедшие в ключевое слово буквы выписываются вслед за ним в алфавитном порядке. Например, ключевое слово "hieronymus" задает матрицу

$$\begin{pmatrix} h & i & e & r & o \\ n & y & m & u & s \\ a & b & c & d & f \\ g & k & l & p & q \\ t & v & w & x & z \end{pmatrix}.$$

Биграмма $(x, y) = (K_{i,j}, K_{m,n})$, в которой $x \neq y$, шифруется как

$$\begin{array}{ll} (K_{i,n}, K_{m,j}), & \text{если } i \neq m \text{ и } j \neq n, \\ (K_{i,j+1}, K_{i,n+1}), & \text{если } i = m \text{ и } j \neq n, \\ (K_{i+1,j}, K_{m+1,j}), & \text{если } i \neq m \text{ и } j = n, \end{array}$$

где вычисление индексов производится по модулю 5. Если же символы x и y в биграмме (x, y) совпадают, то необходимо предварительно вставить между ними букву q и продолжить шифрование текста $...xqy...$.

2.3.3 Шифры перестановок

Принципиально иной способ шифрования называется *перестановкой*. Эта система разбивает текст на блоки одинаковой длины, например, длины n, и применяет к каждому такому блоку фиксированную перестановку σ координат. Например, при $n = 5$ и $\sigma = (1, 4, 5, 2, 3)$ получается следующее шифрование:

$$\text{crypt ograp hical } ... \xrightarrow{\sigma} \text{ytrcp rpgoa cliha } ...$$

Часто перестановка имеет геометрическую природу, как в случае так называемой *столбцовой перестановки*. Открытый текст записывается по строкам в матрицу заданного размера, а читается по столбцам, переставленным в порядке, определяемом ключевым словом. Например, после отождествления букв $a, b, ..., z$ и чисел $1, 2, ..., 26$ ключевое слово "right" указывает, что первым должен читаться 3-й столбец (буква "g" расположена в алфавите раньше других букв слова "right"), за ним 4-й, 2-й, 1-й, и наконец 5-й. Таким образом, открытый текст

computing science has had very little influence on computing practice

при шифровании с помощью матрицы 5×5 и ключа "right" сначала пишется по строкам, как показано ниже,

4	3	1	2	5	4	3	1	2	5	4	3	1	2	5
c	o	m	p	u	y	l	i	t	t	n	g	p	r	a
t	i	n	g	s	l	e	i	n	f	c	t	i	c	e
c	i	e	n	c	l	u	e	n	c	.	.	.		
e	h	a	s	h	e	o	n	c	o					
a	d	v	e	r	m	p	u	t	i					

а затем читается по столбцам в порядке нумерации, в результате чего получается следующий шифртекст:

mneav pgnse oiihd ctcea uschr iienu tnnct leuop yllem tfcoi ...

Перестановки не изменяют частоты букв, но разрушают взаимосвязи между последовательными буквами открытого текста. Шифр Виженера и другие шифры замены действуют прямо противоположным образом. Поэтому часто эти системы комбинируют. Такие комбинированные криптосистемы принято называть *произведениями шифров*. Шеннон для обозначения этих факторов воздействия криптопреобразований на открытый текст употреблял слова "confusion" и "diffusion" ("перемешивание" и "рассеивание").

Криптосистемы, в которых шифрование каждого символа открытого текста производится в зависимости от предыдущих символов, называют *поточными шифрами* (они обсуждаются в гл. 3). Криптосистемы, в которых производится одновременное шифрование целого блока символов фиксированной длины, и шифрования таких блоков независимы друг от друга, называют *блочными шифрами* (они обсуждаются в гл. 4).

Во время Второй мировой войны обе воюющие стороны применяли для шифрования так называемые роторные машины. Опишем основные идеи, заложенные в две из них.

2.3.4 Машина Хагелина

Машина *Хагелина* (или C-36), изобретенная шведским инженером Борисом Хагелином и использовавшаяся армией США, имеет 6 роторов, на которых располагаются штифты в количествах 26, 25, 23, 21, 19 и 17 штук. Каждый из этих штифтов приводится либо в активное, либо в пассивное положение, в зависимости от того, влево или вправо от ротора он выдвигается. После шифрования буквы, которое зависит от установки этих штифтов и положения вращающихся дисков, все 6 роторов поворачиваются на одну позицию. Таким образом, в первоначальном положении первый ротор окажется после шифрования 26-й буквы, а, например, шестой — после 17-й.

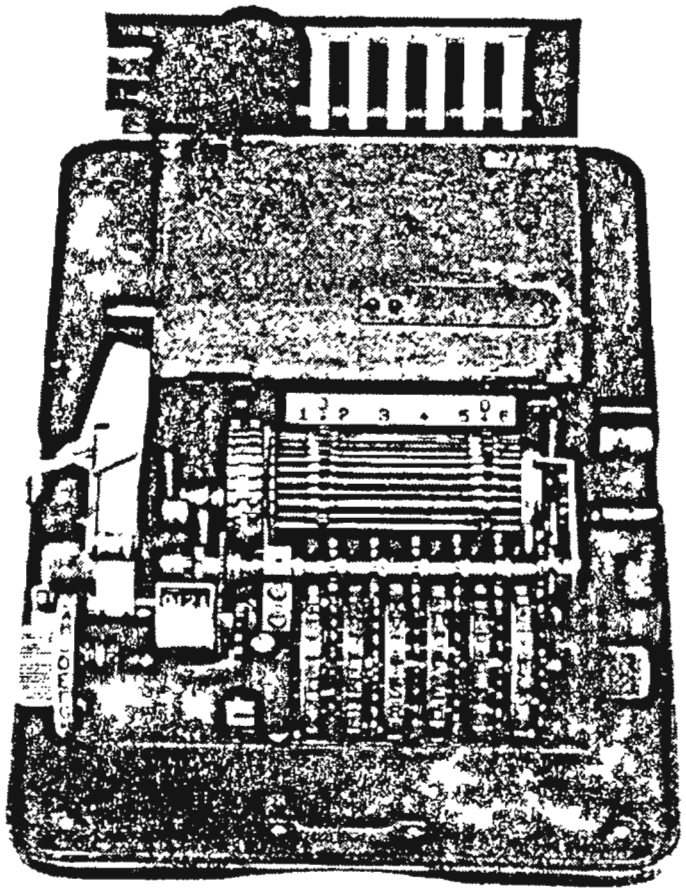


Рис. 2.1. Машина Хагелина.

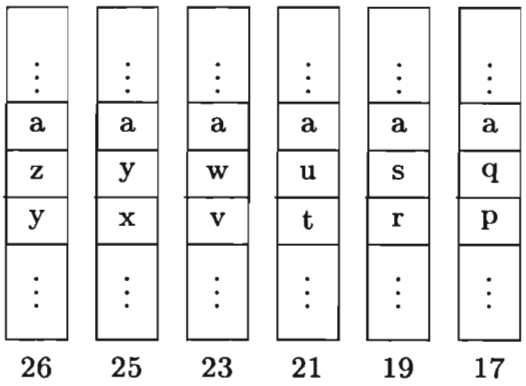


Рис. 2.2. Шесть роторов машины Хагелина, каждый со своим числом позиций.

Поскольку количества штифтов на роторах взаимно просты, можно считать машину Хагелина механическим вариантом криптосистемы Виженера с ключом длины $26 \times 25 \times 23 \times 21 \times 19 \times 17 = 101405850$. Читатель, интересующийся криптоанализом этой машины, может обратиться к разделу 2.3 в [BekP82][2].

[2]Более подробное описание С-36 можно найти в *[Сало96]. — *Прим. перев.*

2.3.5 "Энигма"

Рис. 2.3. "Энигма".

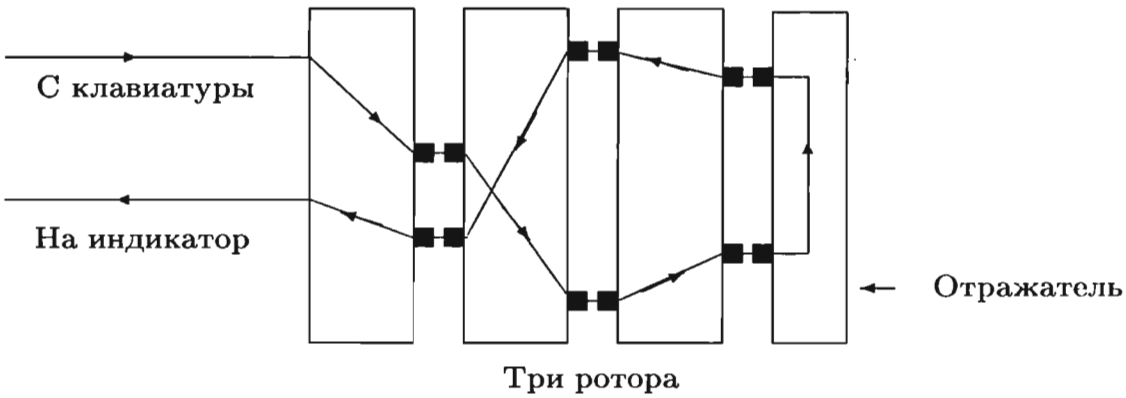


Рис. 2.4. Схематичное описание машины "Энигма".

Электромеханическая машина *"Энигма"* (как называли ее англичане), использовавшаяся Германией и Японией, была изобретена А. Шербиусом в 1923 г. Она состоит из трех подвижных дисков — роторов и одного неподвижного — отражателя (см. рис. 2.4). При нажатии на клавишу с буквой электрический сигнал посылается в ту позицию первого ротора, которая этой букве соответствует. Из ротора сигнал выходит уже в другой позиции, зависящей от устройства электропроводки внутри ротора. Все три ротора делают одно и то же, но устройство электропроводки внутри каждого из них свое. Внутри отражателя ток проходит от одного контакта до какого-то другого, тоже зависящего от внутренней

электропроводки. Затем ток вновь проходит через три ротора в обратном порядке и зажигает лампочку, которая и определяет шифрование исходной буквы.

Одновременно с нажатием клавиши первый ротор поворачивается на одну позицию. После 26 поворотов первого ротора второй поворачивается на одну позицию. После завершения вторым ротором полного поворота на одну позицию повернется третий.

Ключ "Энигмы" состоит из

i) выбора и порядка роторов,

ii) их начального положения,

iii) фиксированной начальной перестановки алфавита.

Идею, на которой основан криптоанализ шифра "Энигмы", можно найти в гл. 5 [Konh81].

2.4 Задачи

Задача 2.1. Следующая криптограмма о президенте Кеннеди (Kennedy) получена с помощью простой замены. Определите открытый текст.

"rgjjg mvkto tzpgt stbgp catjw pgocm gjs"

Задача 2.2. Дешифруйте следующий шифртекст, полученный при использований шифра Плэйфэра с ключом "hieronymous" (как в разд. 2.3.2).

"erohh mfimf ienfa bsesn pdwar gbhah ro"

Задача 2.3. Зашифруйте следующий текст, с помощью шифра Виженера с ключом "vigenere".

"who is afraid of virginia woolf"

Задача 2.4^M. Рассмотрим криптограмму, полученную с помощью шифра Цезаря. Напишите программу в пакете "Mathematica" для поиска всех подстрок длины 5 в шифртексте, которые могли бы быть получены из слова "Brute". Проверьте эту программу на тексте "xyuysuyifvyxi" из табл. 2.1. (См. также входную строку в примере 2.2).

Глава 3

Последовательности регистров сдвига

3.1 Псевдослучайные последовательности

Внедрение логических схем во время и после Второй мировой войны способствовало созданию полностью электронных криптосистем. Они оказались весьма практичными в том смысле, что были легки в реализации и очень быстры. Но анализ их безопасности отнюдь не легок! Работа с логическими схемами часто приводит к алфавиту $\{0,1\}$. Имеются лишь две возможные перестановки (подстановки) на множестве $\{0,1\}$. Одна из них меняет местами оба символа. Это можно описать как прибавление 1 (по модулю 2) к обоим элементам. Другая перестановка оставляет символы неизменными, что означает прибавление 0 (по модулю 2) к этим элементам.

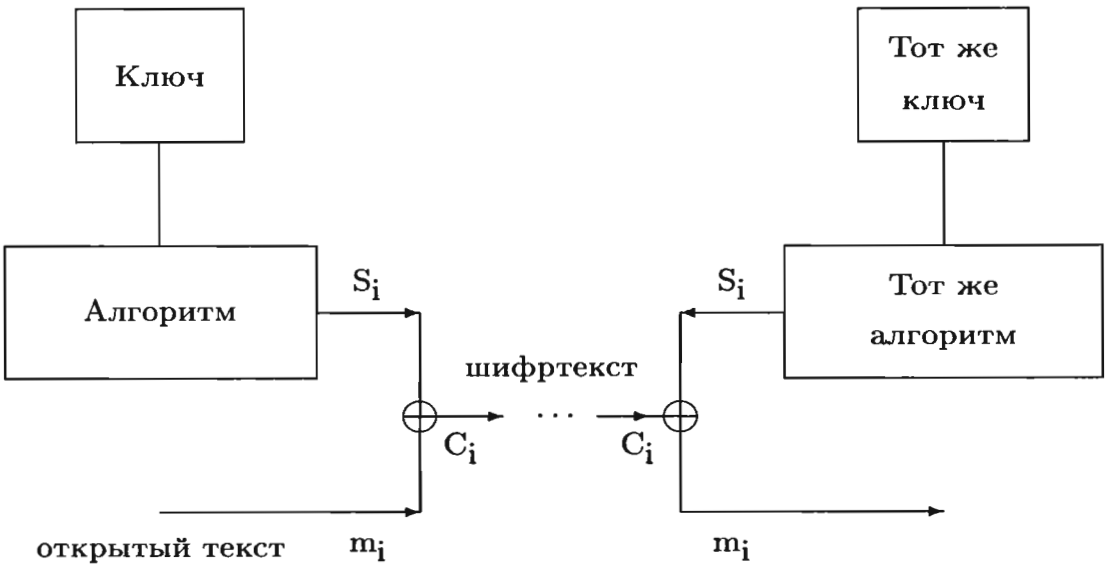


Рис. 3.1. Двоичная криптосистема с псевдослучайной последовательностью $\{s_i\}_{i\geq 0}$.

Так как шифр Вернама безусловно безопасен (абсолютно стоек), но не очень практичен, естественно возникает схема, изображенная на рис.3.1.

Разумеется, желательно, чтобы последовательность $\{s_i\}_{i\geq 0}$ была случайной, но машина с конечным числом состояний и детерминированным алгоритмом работы не может генерировать случайную последователь-

ность. В этой ситуации всегда порождается финально периодическая (ultimately periodic) последовательность (*[ЛидП96]). Это наблюдение показывает, что (исключая начальный сегмент) данная схема есть специальный случай криптосистемы Виженера. С другой стороны, можно попытаться генерировать последовательности, которые кажутся случайными, имеют длинные периоды и хорошие криптографические свойства. Справочными книгами по этой теории могут служить [Bek82], [Gol67] и [Ruep86]. [см. также *[ЛидН88] и *[ЛидП96] — *ред.*]

Голомб [Gol67] сформулировал три постулата, которым должна удовлетворять двоичная периодическая последовательность, чтобы называться *псевдослучайной*. Прежде чем привести их, введем необходимую терминологию.

Определение 3.1
Последовательность $\{s_i\}_{i\geq 0}$ называется *периодической* с *периодом* p, если p — наименьшее натуральное число, для которого

$$s_{i+p} = s_i \qquad \text{при всех } i \geq 0.$$

Отрезок длины k есть подпоследовательность из $\{s_i\}_{i\geq 0}$, состоящая из k идентичных символов, ограниченная иными символами. Если отрезок начинается в момент t, то справедливы условия

$$s_{t-1} \neq s_t = s_{t+1} = \ldots = s_{t+k-1} \neq s_{t+k}.$$

Мы различаем следующие отрезки:

$$\textit{блок длины } k : 0\overbrace{11\ldots1}^{k}0,$$

$$\textit{лакуна длины } k : 1\overbrace{00\ldots0}^{k}1.$$

Автокорреляция $AC(k)$ периодической последовательности $\{s_i\}_{i\geq 0}$ с периодом p определяется формулой:

$$AC(k) = \frac{A(k) - D(k)}{p}, \tag{3.1}$$

где $A(k)$ (соответственно, $D(k)$) обозначает число совпадений (соответственно, несовпадений) на полном периоде между $\{s_i\}_{i\geq 0}$ и $\{s_{i+k}\}_{i\geq 0}$; вторая последовательность есть последовательность $\{s_i\}_{i\geq 0}$, сдвинутая влево на k позиций. Таким образом,

$$A(k) = |\{0 \leq i < p \mid s_i = s_{i+k}\}|,$$

$$D(k) = |\{0 \leq i < p \mid s_i \neq s_{i+k}\}|.$$

Заметим, что можно также написать $AC = (2 \cdot A(k) - p)/p$.

Пример 3.1

Рассмотрим периодическую последовательность с периодом p, заданную ее первыми p элементами.

С помощью функций Count, Length, Mod, RotateLeft и Table пакета "Mathematica" легко вычисляются все значения автокорреляционной функции $AC(k)$, $0 \leq k \leq p-1$.

```
segment = {1, 1, 0, 1, 0, 0, 0, 0};
p = Length[segment];
Table[(2 * Count[Mod[
    segment - RotateLeft[segment, k], 2], 0] - p) / p,
    k, 0, p - 1]
```

```
{1, 0, 0, 0, -1/2, 0, 0, 0}
```

Если k кратно p, получаем $A(k) = p$, $D(k) = 0$, так что $AC = 1$. В этом случае говорят о *внутрифазовой* автокорреляции. Если p не делит k, то говорят о *внефазовой* автокорреляции. В этом случае значение AC лежит между -1 и $+1$.

Определение 3.2 (*Постулаты случайности Голомба*)
G1: Числа нулей и единиц на период равны настолько, насколько это возможно, т.е. оба числа равны $p/2$, если p четно, и равны $(p \pm 1)/2$, если p нечетно.

G2: Половина отрезков в цикле имеют длину 1, четверть отрезков имеют длину 2, восьмая часть отрезков имеют длину 3 и т.д.; кроме того, половина отрезков данной длины — лакуны, другая половина — блоки.

G3: Внефазовая автокорреляция $AC(k)$ имеет одно и то же значение для всех k.

G1 утверждает, что нули и единицы встречаются приблизительно с одной и той же вероятностью. Их появления легко подсчитать с помощью функции *Count* пакета "Mathematica".

```
segment = {1, 1, 0, 1, 0, 0, 0, 0};
Count[segment, 0]
Count[segment, 1]
```

```
5
3
```

G2 влечет, что после комбинации 011 символ 0 (приводящий к блоку длины 2) имеет ту же вероятность, что и символ 1 (приводящий к блоку длины ≥ 3), и т.п. Таким образом, **G2** говорит, что конкретные n-граммы встречаются с правильными частотами. Эти частоты можно вычислить с помощью функций *Count*, *Length*, *Rotateleft*, *Table* и *Take* пакета "Mathematica":

```
segment = {0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
        0, 0, 0, 1, 0, 1, 1};
p = Length[segment];
ngram = {1, 0, 1}; k = Length[ngram];
Count[Table[Take[
    RotateLeft[segment, i], k] == ngram, {i, p}], True]
```

```
3
```

Интерпретация постулата **G3** более трудна. Это условие говорит, что число совпадений в данной последовательности и в ее сдвинутой версии не дает никакой информации о периоде последовательности, если сдвиг не кратен периоду. Похожая ситуация описывается в лемме 3.1, где такое сравнение позволяет определить длину ключа, использованного в шифре Виженера. В криптографических приложениях p слишком велико для такого подхода.

Лемма 3.1
Пусть $\{s_i\}_{i\geq 0}$ — двоичная последовательность с периодом $p > 2$, удовлетворяющая постулатам случайности Голомба. Тогда p нечетно и $AC(k)$ принимает значение $-1/p$, когда k не делится на p.

Доказательство. Рассмотрим циклическую $p \times p$-матрицу с верхней строкой $s_0, s_1, \ldots, s_{p-1}$. Подсчитаем двумя способами число всех совпадений за вычетом числа несовпадений между верхней строкой и всеми прочими строками. В силу постулата **G3** построчный подсчет дает вклад $p \cdot AC(k)$ для каждой строки с номером i, $2 \leq i \leq p$. В целом получается значение $p(p-1)AC(k)$. Теперь мы подсчитаем ту же сумму по столбцам, рассматривая число совпадений за вычетом числа несовпадений между всеми нижними клетками и верхней.

Случай четного p

В силу **G1** вклад каждого столбца равен $(p/2 - 1) - p/2 = -1$, так как каждый столбец содержит в точности $p/2 - 1$ совпадений нижних клеток с верхней и в точности $p/2$ несовпадений. Сумма этих значений по всем столбцам равна $-p$. Приравнивая два значения суммы, получаем $(p-1)AC(k) = -1$. Однако из равенства (3.1) следует, что число $p \cdot AC(k)$ целое. Это невозможно, когда $AC(k) = -1/(p-1)$, так как $p > 2$.

Случай нечетного p

Для $(p+1)/2$ столбцов получаем вклад $(p-1)/2 - (p-1)/2$, равный 0, а для $(p-1)/2$ столбцов — вклад $(p-3)/2 - (p+1)/2$, равный -2. Следовательно, значение суммы есть $-(p-1)$. Приравнивание этой суммы к $p(p-1)AC(k)$ дает $AC(k) = -1/p$.

■

Хорошо известный критерий χ^2 и спектральный критерий [CovM67] дают способы проверки свойств псевдослучайности данной последовательности. Мы не будем обсуждать здесь эти методы. Заинтересованный

читатель отсылается к [Golo67], гл. IV, [Knut81] или [Knut69], гл. 3, или же к универсальному статистическому критерию Маурера [Maur92].

Имеются также свойства криптографической природы, которым должна удовлетворять последовательность $\{s_i\}_{i\geq 0}$ на рис. 3.1.

C1: период последовательности $\{s_i\}_{i\geq 0}$ должен быть очень большим (величиной порядка 10^{50});

C2: последовательность $\{s_i\}_{i\geq 0}$ должна легко генерироваться;

C3: знание части открытого текста и соответствующего шифртекста (атака с известным открытым текстом) не должно позволить криптоаналитику сгенерировать всю последовательность $\{s_i\}_{i\geq 0}$.

3.2 Линейные регистры сдвига с обратной связью

3.2.1 (Линейные) регистры сдвига с обратной связью

Регистры сдвига с обратной связью реализуют очень быстрый способ порождения бинарных (или двоичных) последовательностей. Их общий вид показан на рис. 3.2.

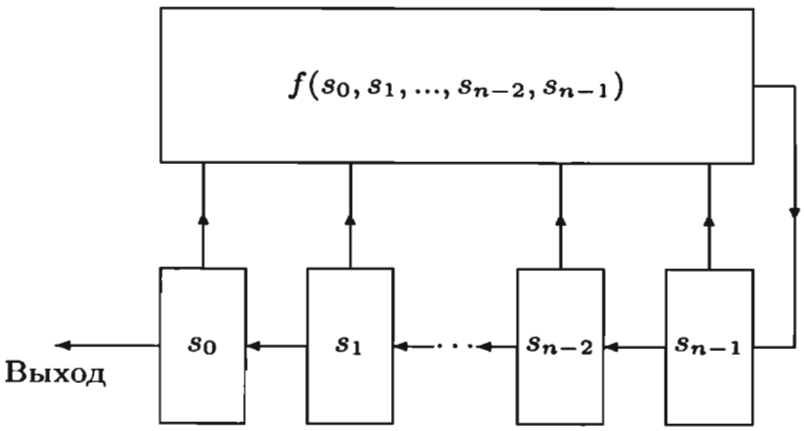


Рис. 3.2. Общий вид регистра сдвига с обратной связью.

Регистр сдвига с обратной связью (FSR — от feedback shift register) *длины* n имеет n ячеек памяти, значения которых совместно образуют (начальное) *состояние* $(s_0, s_1, \ldots, s_{n-1})$ регистра сдвига. Функция f отображает $\{0,1\}^n$ в $\{0,1\}$ и называется *функцией обратной связи* регистра. Так как f является булевой функцией, она может быть легко построена из элементарных логических функций.

После первого такта работы регистр сдвига выдаст s_0 и перейдет в состояние $(s_1, s_2, \ldots, s_n)$ где $s_n = f(s_0, s_1, \ldots, s_{n-1})$. Продолжая таким образом, регистр сдвига генерирует бесконечную последовательность $\{s_i\}_{i\geq 0}$.

Пример 3.2

Рассмотрим случай $n = 3$, *когда* f *задается равенством* $f(s_0, s_1, s_2) = s_0 s_1 + s_2$. *Исходя из начального состояния* (s_0, s_1, s_2), *можно легко определить последующие состояния с помощью функций* Mod, Do *и* Print *пакета "Mathematica"*:

```
Clear[f];
f[x_, y_, z_] := Mod[x*y + z, 2];
{s0, s1, s2} = {0, 1 ,1};
Do[ {s0, s1, s2} = {s1, s2, f[s0, s1, s2]};
    Print[{s0, s1, s2}], {i, 1, 6}]
```

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

В этом разделе мы изучим специальный случай *линейной* функции f:

$$f(s_0, s_1, \ldots, s_{n-1}) = c_0 s_0 + c_1 s_1 + \ldots + c_{n-1} s_{n-1},$$

где все c_i — двоичные цифры, а сложение выполняется по модулю 2. Общий вид *линейного регистра сдвига с обратной связью* (кратко LFSR) показан на рис. 3.3.

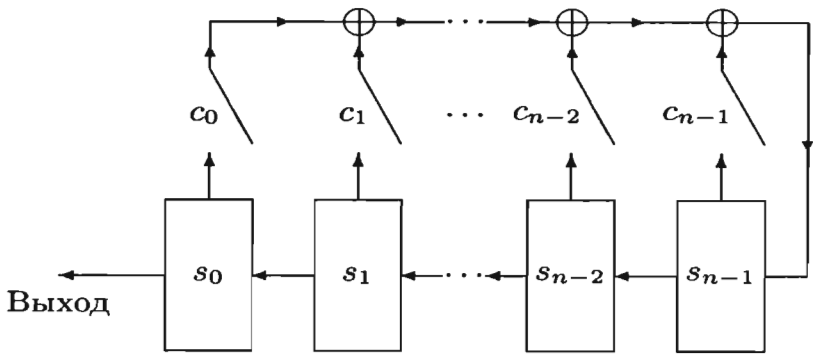


Рис. 3.3. Общий вид линейного регистра сдвига.

Выходная последовательность $\{s_i\}_{i \geq 0}$ такого LFSR может быть описана посредством начального состояния $(s_0, s_1, \ldots, s_{n-1})$ и линейного рекуррентного соотношения

$$s_{k+n} = \sum_{i=0}^{n-1} c_i s_{k+i}, \qquad k \geq 0, \tag{3.2}$$

или, эквивалентно,

$$\sum_{i=0}^{n} c_i s_{k+i} = 0, \qquad k \geq 0, \tag{3.3}$$

где $c_n = 1$ по определению. Пусть $\underline{s}^{(i)}$ обозначает состояние в момент i, т.е. $\underline{s}^{(i)} = (s_i, s_{i+1}, \ldots, s_{i+n-1})$. Имеется следующее, аналогичное (3.2), рекуррентное соотношение для последовательных состояний регистра:

$$\underline{s}^{(k+n)} = \sum_{i=0}^{n-1} c_i \underline{s}^{(k+i)}, \qquad k \geq 0. \tag{3.4}$$

Коэффициенты c_i в (3.2) и на рис. 3.3 называются *коэффициентами обратной связи* регистра. Если $c_i = 0$, то соответствующий переключатель на рис. 3.3 разомкнут, а если $c_i = 1$, то замкнут. Мы всегда будем предполагать, что $c_0 = 1$. Следствием данного предположения является то, что любое состояние LFSR имеет не только единственное следующее состояние, что естественно, но и единственного предшественника. В самом деле, для любого $k \geq 0$ значение s_k однозначно определяется по $s_{k+1}, \ldots, s_{k+n}$ посредством (3.2). Позже (в теореме 3.22) мы докажем это свойство в более общей ситуации.

Пример 3.3

При $n = 4, c_0 = c_1 = 1, c_2 = c_3 = 0$ *мы получаем LFSR на рис.* 3.4.

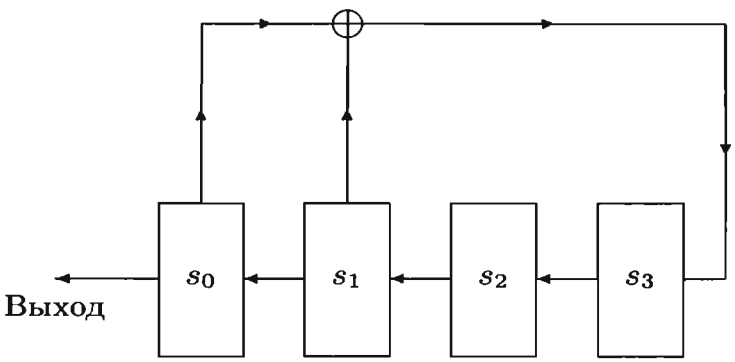


Рис. 3.4. Пример LFSR при $n = 4$.

Начальное состояние $(1, 0, 0, 0)$ *дает следующий список последовательных состояний:*

```
{s0, s1, s2, s3} = {1, 0, 0, 0};
Do[{s0, s1, s2, s3} = {s1, s2, s3, Mod[s0 + s1, 2]};
   Print[i, " ", {s0, s1, s2, s3}], {i, 15}]
```

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

 1   {0, 0, 0, 1}      6   {0, 1, 1, 0}     11   {0, 1, 1, 1}
 2   {0, 0, 1, 0}      7   {1, 1, 0, 1}     12   {1, 1, 1, 1}
 3   {0, 1, 0, 0}      8   {1, 0, 1, 0}     13   {1, 1, 1, 0}
 4   {1, 0, 0, 1}      9   {0, 1, 0, 1}     14   {1, 1, 0, 0}
 5   {0, 0, 1, 1}     10   {1, 0, 1, 1}     15   {1, 0, 0, 0}
```

Заметим, что состояние в момент $t = 15$ *идентично состоянию в момент* $t = 0$, *так что выходная последовательность* $\{s_i\}_{i\geq 0}$ *имеет период* 15.

Выходную последовательность регистра можно легко определить с помощью функций *Table*, *Mod* и *Do* пакета "Mathematica":

```
Clear[s]; {s[0], s[1], s[2], s[3]} = {1, 0, 0, 0};
s[j_] := Mod[s[j-4] + s[j-3], 2];
Table[s[j], {j, 0, 15}]
```

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

Состояние называется *нулевым*, если все его компоненты равны 0. Так как для LFSR длины n существуют в точности $2^n - 1$ ненулевых состояний, а нулевое состояние всегда переходит само в себя, можно заключить, что период выходной последовательности никогда не превосходит $2^n - 1$.

3.2.2 PN-последовательности

Определение 3.3
PN-последовательность — это выходная последовательность LFSR длины n, имеющая период $2^n - 1$.

Если LFSR длины n не пробегает циклически все $2^n - 1$ ненулевых состояний, то он определенно не порождает PN-последовательности[1]. В качестве следствия получаем следующее утверждение.

Лемма 3.2
LFSR длины n, порождающий PN-последовательность $\{s_i\}_{i\geq 0}$, циклически пробегает все $2^n - 1$ ненулевых состояний. Любая ненулевая выходная последовательность этого регистра есть сдвиг последовательности $\{s_i\}_{i\geq 0}$.

Мы хотим классифицировать все LFSR, порождающие PN-последовательности. С этой целью мы сопоставим регистру с коэффициентами обратной связи $c_0, c_1, \ldots, c_{n-1}$ его *характеристический многочлен* $f(x)$, определяемый следующим образом:

$$f(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} + x^n = \sum_{i=0}^{n} c_i x^i, \qquad (3.5)$$

где $c_n = 1$ по определению и $c_0 = 1$ по предположению.

[1] PN — от "pseudo-noise". К сожалению, перевод этого слова на русский язык "псевдослучайная", как и для "pseudo-random" из предыдущего подраздела. В русской литературе PN-последовательности обычно называют "последовательностями максимального периода".— *Прим. ред.*

Определение 3.4
Пусть $f = \sum_{i=0}^{n} c_i x^i$. Тогда

$$\Omega(f) = \left\{ \{s_i\}_{i\geq 0} \middle| \{s_i\}_{i\geq 0} \quad \text{удовлетворяет (3.2)} \right\}.$$

Иначе говоря, $\Omega(f)$ есть множество всех выходных последовательностей LFSR с характеристическим многочленом $f(x)$.

Лемма 3.3.
Пусть f — характеристический многочлен LFSR длины n. Тогда $\Omega(f)$ — бинарное векторное пространство размерности n.

Доказательство. Так как соотношение (3.2) линейно, очевидно, что $\Omega(f)$ — векторное пространство. Каждая $\{s_i\}_{i\geq 0}$ из $\Omega(f)$ однозначно определяется первыми n членами $s_0, s_1, \ldots, s_{n-1}$ (начальным состоянием), так что размерность $\Omega(f)$ не больше n. С другой стороны, n различных последовательностей, начинающихся с

$$\overbrace{00\ldots0}^{i}1\overbrace{00\ldots0}^{n-i-1},$$

$0 \leq i \leq n-1$, очевидно, линейно независимы. Поэтому размерность $\Omega(f)$ не меньше n.

■

Пусть f — многочлен степени n, скажем, $f(x) = \sum_{i=0}^{n} c_i x^i$, где $c_n \neq 0$. *Взаимный многочлен* f^* определяется равенством

$$f^*(x) = x^n f(1/x) = c_0 x^n + c_1 x^{n-1} + \ldots + c_{n-1} x + c_n = \sum_{i=0}^{n} c_{n-i} x^i. \quad (3.6)$$

Сопоставим последовательности $\{s_i\}_{i\geq 0}$ степенной ряд, называемый *порождающей функцией* последовательности:

$$S(x) = \sum_{i=0}^{\infty} s_i x^i. \quad (3.7)$$

Вместо записи $\{s_i\}_{i\geq 0} \in \Omega(f)$ будем также использовать запись $S(x) \in \in \Omega(f)$. Мы знаем, что $S(x)$ однозначно определяется начальным состоянием $(s_0, s_1, \ldots, s_{n-1})$ и характеристическим многочленом $f(x)$. В следующей теореме эта зависимость делается более явной.

Теорема 3.4
Пусть $\{s_i\}_{i\geq 0} \in \Omega(f)$, где f задан равенством (3.5), а $S(x)$ — порождающая функция последовательности $\{s_i\}_{i\geq 0}$. Тогда $S(x)f^*(x)$ — многочлен степени, меньшей n.

Доказательство.

$$S(x)f^*(x) \overset{(3.6)\&(3.7)}{=} \left(\sum_{k=0}^{\infty} s_k x^k\right)\left(\sum_{l=0}^{n} c_{n-l} x^l\right) =$$

$$= \sum_{j=0}^{\infty} \left(\sum_{l=0}^{\min\{j,n\}} c_{n-l} s_{j-l} \right) x^j =$$
$$= \sum_{j=0}^{n-1} \left(\sum_{l=0}^{j} c_{n-l} s_{j-l} \right) x^j + \sum_{j=n}^{\infty} \left(\sum_{l=0}^{n} c_{n-l} s_{j-l} \right) x^j =$$
$$= \sum_{j=0}^{n-1} \left(\sum_{l=0}^{j} c_{n-l} s_{j-l} \right) x^j + \sum_{j=n}^{\infty} \left(\sum_{i=0}^{n} c_i s_{(j-n)+i} \right) x^j =$$
$$\overset{(3.3)}{=} \sum_{j=0}^{n-1} \left(\sum_{l=0}^{j} c_{n-l} s_{j-l} \right) x^j.$$

■

Замечание. Из приведенного доказательства следует, что $S(x) = u(x)/f^*(x)$, где $u(x) = \sum_{j=0}^{n-1} \left(\sum_{l=0}^{j} c_{n-l} s_{j-l} \right) x^j$ — многочлен степени меньше n, коэффициенты которого зависят от начального состояния и характеристического многочлена. Отметим также, что отображение $S(x) \mapsto S(x) f^*(x)$ инъективно, так как $f^*(x) \neq 0$.

Пример 3.4

Рассмотрим LFSR с $n = 5, f(x) = 1 + x^2 + x^5$ *и возьмем в качестве начального состояние* $(1, 1, 0, 1, 0)$. *Многочлен* $u(x)$ *можно вычислить с помощью функции PolynomialMod:*

```
{c[0], c[1], c[2], c[3], c[4], c[5]} = {1, 0, 1, 0, 0, 1};
{s[0], s[1], s[2], s[3], s[4]} = {1, 1, 0, 1, 0};
u = PolynomialMod[∑_{j=0}^{4} ∑_{l=0}^{j} c[5-l]s[j-l]x^j, 2]
```

|| `1 + x + x^4`

Чтобы проверить теорему 3.4 *до некоторого члена* x^L *в* $S(x)$, *воспользуемся соотношением* (3.2) *для вычисления всех* s_i *вплоть до* L*-го. Здесь применяются функции Mod, Print и PolynomialMod:*

```
{c[0], c[1], c[2], c[3], c[4], c[5]} = {1, 0, 1, 0, 0, 1};
{s[0], s[1], s[2], s[3], s[4]} = {1, 1, 0, 1, 0};
fstar = ∑_{i=0}^{5} c[5-i]x^i;
L = 60;
s[i_] := s[i] = Mod[s[i-5] + s[i-3], 2];
s = ∑_{i=0}^{L} s[i]x^i; Print[s];
PolynomialMod[S*fstar, {x^L, 2}]
```

$1+x+x^3+x^5+x^{10}+x^{13}+x^{15}+x^{16}+x^{19}+x^{20}+x^{21}+x^{22}+x^{23}+x^{27}+x^{28}+$
$+x^{30}+x^{31}+x^{32}+x^{34}+x^{36}+x^{41}+x^{44}+x^{46}+x^{47}+x^{50}+x^{51}+x^{52}+x^{53}+$
$+x^{54}+x^{58}+x^{59}$

|| `1 + x + x^4`

Выход действительно тот же, что и выше.

Следствие 3.5

$$\Omega(f) = \left\{ \frac{u(x)}{f^*(x)} \middle| \operatorname{degree}(u(x)) < n \right\}.$$

Замечание. Запись $S(x) = u(x)/f^*(x)$ означает то же самое, что и $S(x)f^*(x) = u(x)$.

Доказательство. Из теоремы 3.4 и замечания выше следует, что каждый член из $\Omega(f)$ можно записать в виде $u(x)/f^*(x)$ с $\operatorname{degree}(u(x)) < n$, причем многочлен $u(x)$ — единственный. Это доказывает включение $\subseteq$.

С другой стороны, по лемме 3.3 мощность $\Omega(f)$ равна 2^n, и имеется также в точности 2^n бинарных многочленов $u(x)$ степени $< n$.

■

Теперь легко доказать следующую лемму. Пусть $\text{НОК}[f, g]$ обозначает наименьшее общее кратное многочленов f и g.

Лемма 3.6
Пусть f и g — (характеристические) многочлены, $\{s_i\}_{i\geq 0} \in \in \Omega(f)$, $\{t_i\}_{i\geq 0} \in \Omega(g)$. Тогда

$$\{s_i + t_i\}_{i\geq 0} \in \Omega(\text{НОК}[f, g]).$$

Доказательство. Положим $h = \text{НОК}[f, g]$ и $h = a \cdot f = b \cdot g$. Пусть $S(x)$ и $T(x)$ — порождающие функции последовательностей $\{s_i\}_{i\geq 0}$ и $\{t_i\}_{i\geq 0}$ соответственно.

Следствие 3.5 влечет, что $S(x) = u(x)/f^*(x)$ и $T(x) = v(x)/g^*(x)$, где $\operatorname{degree}(u(x)) < \operatorname{degree}(f(x))$ и $\operatorname{degree}(v(x)) < \operatorname{degree}(g(x))$. Поскольку

$$S(x) + T(x) = \frac{u(x)}{f^*(x)} + \frac{v(x)}{g^*(x)} = \frac{a^*(x)u(x)}{a^*(x)f^*(x)} + \frac{b^*(x)v(x)}{b^*(x)g^*(x)} =$$

$$= \frac{a^*(x)u(x) + b^*(x)v(x)}{h^*(x)}$$

и как $a^*(x)u(x)$, так и $b^*(x)v(x)$ имеют степени меньше, чем $\operatorname{degree}(h(x))$, получаем $S(x) + T(x) \in \Omega(h)$.

■

3.2.3 Какие характеристические многочлены задают PN-последовательности?

Периодом[2] многочлена f при условии $f(0) \neq 0$ называют наименьшее положительное m, такое, что $f(x)$ делит $x^m - 1$, т.е. наименьшее положительное m со свойством $x^m \equiv 1 \pmod{f(x)}$. Оно вполне определено,

[2]Обычно это число называют *порядком* (order) многочлена.— *Прим. ред.*

ибо последовательность степеней x, приведенных по модулю $f(x)$, является периодической. В самом деле, если $x^i \equiv x^j \pmod{f(x)}$ и $0 < i < j$, то также $x^{i-1} \equiv x^{j-1} \pmod{f(x)}$, поскольку НОК$(x, f(x)) = 1$. (Одночлен x имеет мультипликативный обратный по следствию B.14, поэтому можно делить на x.) Повторяем этот процесс, пока не получим $1 \equiv x^{j-i} \pmod{f(x)}$.

Пример 3.5

Пусть $f(x) = 1 + x^4 + x^5$. Его период можно вычислить с помощью функций While и PolynomialMod пакета "Mathematica". Именно, начиная с x (испытывая $m = 1$), вычисляем последовательные степени x, умножая предыдущую степень на x (что соответствует циклическому сдвигу), затем приводим результат по модулю $f(x)$, пока не получим на выходе 1.

```
f = 1 + x^4 + x^5; m = 1; u = x;
While[u != 1, u = PolynomialMod[
        x*u, {f, 2}; m = m + 1];
m
```

```
21
```

Из теоремы B.35 следует, что бинарный неприводимый многочлен степени n делит $x^{2^n-1} - 1$, а потому период m такого многочлена делит $2^n - 1$. (Это наблюдение можно использовать для более эффективного вычисления периода многочлена. Однако мы не будем сейчас обсуждать эту технику. См. конец примера 8.2.)

Лемма 3.7
Пусть $\{s_i\}_{i\geq 0} \in \Omega(f)$, где f — многочлен степени n и периода m, причем $f(0) \neq 0$. Тогда период последовательности $\{s_i\}_{i\geq 0}$ делит m.

Доказательство. Запишем $x^m - 1 = f(x)g(x)$. Переход к взаимным многочленам дает $x^m - 1 = f^*(x)g^*(x)$. По следствию 3.5 существует такой многочлен $u(x)$ степени меньше n, что

$$S(x) = \frac{u(x)}{f^*(x)} = \frac{u(x)g^*(x)}{f^*(x)g^*(x)} = \frac{u(x)g^*(x)}{1 - x^m} = u(x)g^*(x)(1 + x^m + x^{2m} + \ldots).$$

Так как degree$(u(x)g^*(x)) <$ degree$(f^*(x)g^*(x)) =$ degree$(x^m - 1) = m$, ясно, что период $S(x)$ должен делить m.

■

Лемма 3.8
Пусть $\{s_i\}_{i\geq 0} \in \Omega(f)$, где f — неприводимый многочлен степени n и периода m. Тогда последовательность $\{s_i\}_{i\geq 0}$ также имеет период m.

Доказательство. Пусть $\{s_i\}_{i\geq 0}$ имеет период p. По лемме 3.7 p делит m. Положим $S^{(p)}(x) = \sum_{i=0}^{p-1} s_i x^i$. Отсюда

$$S(x) = S^{(p)}(x)(1 + x^p + x^{2p} + \ldots) = S^{(p)}(x)/(1 - x^p).$$

С другой стороны, $S(x) = u(x)/f^*(x)$ по следствию 3.5. Приравнивая эти выражения, получаем

$$S^{(p)}(x)f^*(x) = u(x)(x^p - 1),$$

откуда

$$(S^{(p)}(x))^* f(x) = u^*(x)(x^p - 1).$$

Так как $f(x)$ неприводим и имеет степень n, а $\mathrm{degree}(u(x)) < n$, получаем, что $f(x)$ делит $x^p - 1$. Поэтому период m многочлена $f(x)$ должен делить p. Таким образом, $p = m$.

■

Пример 3.6

Рассмотрим неприводимый многочлен $f(x) = 1 + x + x^2 + x^3 + x^4$, *имеющий период* 5, *так как* $(x-1)f(x) = x^5 - 1$. *По лемме* 3.8 *выходные последовательности из* $\Omega(f)$ *также имеют период* 5, *что может быть легко проверено*:

```
{s0, s1, s2, s3} = {1, 1, 0, 0};
Do[{s0, s1, s2, s3} = {s1, s2, s3, Mod[s0 + s1 + s2 + s3, 2]};
      Print[i, " ", {s0, s1, s2, s3}], {i, 5}]
```

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

1    {1, 0, 0, 0}        4    {0, 1, 1, 0}
2    {0, 0, 0, 1}        5    {1, 1, 0, 0}
3    {0, 0, 1, 1}
```

Есть окольный путь нахождения неприводимого многочлена степени n — разложить $x^{2^n-1} - 1$ с помощью функции *Factor* из "Mathematica":

```
n = 5;
Factor[x^(2^n-1) - 1, Modulus -> 2]
```

$$(1+x)(1+x^2+x^5)(1+x^3+x^5)(1+x+x^2+x^3+x^5)$$
$$(1+x+x^2+x^4+x^5)(1+x+x^3+x^4+x^5)(1+x^2+x^3+x^4+x^5)$$

В пакете "Mathematica" можно найти неприводимый многочлен над $\mathbb{F}_p$, p простое, с помощью функции *IrreduciblePolynomial*, для чего предварительно загружается пакет *Algebra`FiniteFields`*.

```
<< Algebra`FiniteFields`
```

```
p = 2; deg = 11;
IrreduciblePolynomial[x, p, deg]
```

$$1 + x^9 + x^{11}$$

Лемма 3.9
Пусть $\{s_i\}_{i\geq 0}$ - PN-последовательность, порожденная LFSR с характеристическим многочленом f. Тогда f неприводим.

Доказательство. Запишем $f = f_1 f_2$, где f_1 неприводим и имеет степень $n_1 > 0$. По следствию 3.5 $1/f_1^*(x) \in \Omega(f_1)$, так что по лемме 3.7 и теореме B.35 период последовательности $1/f_1^*(x)$ делит $2^{n_1} - 1$.

С другой стороны, $1/f_1^*(x) = f_2^*(x)/f^*(x) \in \Omega(f)$ и по лемме 3.2 $1/f_1^*(x)$ - циклический сдвиг последовательности $\{s_i\}_{i\geq 0}$ и поэтому имеет период $2^n - 1$. Это возможно, только если $n = n_1$, т.е. если $f(x)$ равен неприводимому делителю $f_1(x)$.

■

Пример 3.7

Рассмотрим $f(x) = (1 + x + x^2)(1 + x + x^3) = 1 + x^4 + x^5$. *Легко проверить, что* $1+x+x^2$ *делит* x^3-1, *а* $1+x+x^3$ *делит* x^7-1. *Поскольку числа* 3 *и* 7 *взаимно просты,* $f(x)$ *делит* $x^{21} - 1$. *Отсюда следует, что период каждой выходной последовательности делит* 21. *Это можно проверить для различных начальных состояний следующим образом.*

```
{s0, s1, s2, s3, s4} = {1, 0, 0, 0, 0};
Do[{s0, s1, s2, s3, s4} = {s1, s2, s3, s4, Mod[s0 + s4, 2]};
      Print[i, " ", {s0, s1, s2, s3, s4}], {i, 21}]
```

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

 1   {0, 0, 0, 0, 1}        12   {0, 1, 0, 0, 1}
 2   {0, 0, 0, 1, 1}        13   {1, 0, 0, 1, 1}
 3   {0, 0, 1, 1, 1}        14   {0, 0, 1, 1, 0}
 4   {0, 1, 1, 1, 1}        15   {0, 1, 1, 0, 0}
 5   {1, 1, 1, 1, 1}        16   {1, 1, 0, 0, 0}
 6   {1, 1, 1, 1, 0}        17   {1, 0, 0, 0, 1}
 7   {1, 1, 1, 0, 1}        18   {0, 0, 0, 1, 0}
 8   {1, 1, 0, 1, 0}        19   {0, 0, 1, 0, 0}
 9   {1, 0, 1, 0, 1}        20   {0, 1, 0, 0, 0}
10   {0, 1, 0, 1, 0}        21   {1, 0, 0, 0, 0}
11   {1, 0, 1, 0, 0}
```

Читатель может испытать начальное состояние $(1, 1, 1, 0, 0)$ *и посмотреть, каков период выходной последовательности. Эта последова-*

тельность порождается также LFSR с характеристическим многочленом $1 + x + x^3$ *и начальным состоянием* $(1, 1, 1)$.

Теперь мы готовы доказать главный результат этого подраздела. Напомним читателю определение примитивного многочлена f (степени n): это неприводимый многочлен с тем свойством, что x является примитивным элементом в $\mathrm{GF}(2)[x]/(f(x))$. Это непосредственно переводится в эквивалентное свойство: $f(x)$ имеет период $2^n - 1$.

Теорема 3.10
Ненулевая выходная последовательность LFSR с характеристическим многочленом $f(x)$ тогда и только тогда является PN-последовательностью, когда многочлен $f(x)$ примитивен.

Доказательство. Пусть $f(x)$ имеет степень n.

$\Longrightarrow$ Пусть $\{s_i\}_{i\geq 0} \in \Omega(f)$ является PN-последовательностью. Из леммы 3.9 следует, что $f(x)$ неприводим. В свою очередь, лемма 3.8 влечет, что $f(x)$ должен иметь период $2^n - 1$, что означает его примитивность.

$\Longleftarrow$ Если $f(x)$ примитивен, то он неприводим. По лемме 3.8 $\{s_i\}_{i\geq 0}$ имеет тот же период, что и $f(x)$, т.е. $2^n - 1$. Отсюда следует, что $\{s_i\}_{i\geq 0}$ PN-последовательность.

■

"Mathematica" находит примитивный многочлен степени n над $\mathbb{F}_p$ посредством функции *FieldIrreducible*.

```
n = 5; p = 2;
FieldIrreducible[GF[p, n], x]
```

```
|| 1 + x^3 + x^5
```

Проверим, что этот многочлен действительно определяет PN-последовательность.

```
{s0, s1, s2, s3, s4} = {1, 0, 0, 0, 0};
Do[{s0, s1, s2, s3, s4} = {s1, s2, s3, s4, Mod[s0 + s3, 2]};
     Print[i, " ", {s0, s1, s2, s3, s4}], {i, 31}]
```

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

```
 1   {0, 0, 0, 0, 1}        17   {0, 1, 1, 1, 1}
 2   {0, 0, 0, 1, 0}        18   {1, 1, 1, 1, 1}
 3   {0, 0, 1, 0, 1}        19   {1, 1, 1, 1, 0}
 4   {0, 1, 0, 1, 0}        20   {1, 1, 1, 0, 0}
 5   {1, 0, 1, 0, 1}        21   {1, 1, 0, 0, 1}
 6   {0, 1, 0, 1, 1}        22   {1, 0, 0, 1, 1}
 7   {1, 0, 1, 1, 1}        23   {0, 0, 1, 1, 0}
 8   {0, 1, 1, 1, 0}        24   {0, 1, 1, 0, 1}
 9   {1, 1, 1, 0, 1}        25   {1, 1, 0, 1, 0}
10   {1, 1, 0, 1, 1}        26   {1, 0, 1, 0, 0}
```

```
11   {1, 0, 1, 1, 0}        27   {0, 1, 0, 0, 1}
12   {0, 1, 1, 0, 0}        28   {1, 0, 0, 1, 0}
13   {1, 1, 0, 0, 0}        29   {0, 0, 1, 0, 0}
14   {1, 0, 0, 0, 1}        30   {0, 1, 0, 0, 0}
15   {0, 0, 0, 1, 1}        31   {1, 0, 0, 0, 0}
16   {0, 0, 1, 1, 1}
```

Чтобы найти все примитивные многочлены степени n, можно разложить многочлен $Q^{(2^n-1)}(x)$ деления круга (см. определение В.19). С помощью функций *Factor* и *Cyclotomic* пакета "Mathematica" это достигается следующим образом.

```
p = 2; m = 6; n = p^m - 1;
Factor[Cyclotomic[n, x], Modulus -> 2]
```

$$(1+x+x^6)(1+x+x^3+x^4+x^6)(1+x^5+x^6)$$
$$(1+x+x^2+x^5+x^6)(1+x^2+x^3+x^5+x^6)(1+x+x^4+x^5+x^6)$$

Приводимое ниже следствие прямо вытекает из теорем 3.10 и В.40.

Следствие 3.11
Существуют $\varphi(2^n-1)/n$ различных LFSR длины n, порождающих PN-последовательности. Здесь φ — функция Эйлера (определение А.6).

Более или менее экспоненциальный рост функции $\varphi(2^n-1)/n$ уже для умеренных значений n не дает криптоаналитику возможности угадать правильный примитивный многочлен или исчерпывающе проверить их все. С помощью функции *EulerPhi* пакета "Mathematica" это легко увидеть.

```
n = 100;
EulerPhi[2^n - 1]/n
```

```
570767634000000000000000000
```

3.2.4 Альтернативное описание $\Omega(f)$ для неприводимого f

Мы решим рекуррентное соотношение (3.2) в случае, когда соответствующий характеристический многочлен $f(x) = \sum_{i=0}^{n} c_i x^i$ неприводим. Это включает, конечно, случай примитивного f, для которого, как мы знаем, соответствующий LFSR выдает PN-последовательность.

Последуем стандартному математическому методу решения линейных рекуррентных соотношений. Подстановка $s_j = A \cdot \alpha^j, j \geq 0$, в $s_{k+n} = \sum_{i=0}^{n-1} c_i s_{k+i}$ приводит к уравнению

$$A \cdot \alpha^{k+n} = \sum_{i=0}^{n-1} c_i \cdot A \cdot \alpha^{k+i}.$$

Здесь A и α — элементы расширения поля GF(2); это расширение мы скоро введем. Разделив наше уравнение на $A \cdot \alpha^k$, получим $\alpha^n = \sum_{i=0}^{n-1} c_i \alpha^i$, т.е.

$$f(\alpha) = 0.$$

Изучим более детально случай неприводимого f. Поле Галуа $\mathrm{GF}(2^n) = \mathrm{GF}(2)[x]/(f(x))$ (см. теорему B.16) содержит корень многочлена f. Обозначив его через α, заметим, что

$$\mathrm{GF}(2^n) = \left\{ \sum_{i=0}^{n-1} a_i \alpha^i \middle| a_i \in \mathrm{GF}(2), 0 \le i < n \right\}$$

вместе с обычным покоэффициентным сложением и регулярным правилом умножения (см. (B.3) и (B.4)), но степени α с показателями $\ge n$ всегда приводятся к выражениям степени $< n$ посредством соотношения $\alpha^n = \sum_{i=0}^{n-1} c_i \alpha^i$ (как показано в примере B.5, где вместо символа α использована буква x).

Пример 3.8

Рассмотрим $f(x) = 1+x+x^4$, *и пусть* α — *корень* $f(x)$, *так что* $\alpha^4 = 1 + \alpha$. *Сложение элементов* $1 + \alpha + \alpha^3$ *и* $\alpha + \alpha^2$ *в* $\mathrm{GF}(2)[x]/(f(x))$ *дает* $1 + \alpha^2 + \alpha^3$. *Их умножение дает* $\alpha + \alpha^3 + \alpha^4 + \alpha^5$, *что равно* $(\alpha+1)f(\alpha)+(1+\alpha+\alpha^2+\alpha^3)$; *поэтому результатом будет* $1+\alpha+\alpha^2+\alpha^3$. *Все это можно вычислить с помощью функции* PolynomialMod *пакета "Mathematica"*:

```
f = 1 + a + a^4;
PolynomialMod[(1 + a + a^3) + (a + a^2), {f, 2}]
PolynomialMod[(1 + a + a^3) * (a + a^2), {f, 2}]
```

```
1 + a^2 + a^3
1 + a + a^2 + a^3
```

Лемма 3.12
Пусть f — бинарный неприводимый многочлен степени n и α — корень f в $\mathrm{GF}(2^n)$. Далее, пусть L — нетривиальное линейнос отображение из $\mathrm{GF}(2^n)$ в GF(2). Тогда

$$\Omega(f) = \left\{ \{L(A \cdot \alpha^j)\}_{i \ge 0} \middle| A \in \mathrm{GF}(2^n) \right\}.$$

Доказательство. Мы должны проверить несколько фактов.

i) Последовательность $\{s_j\}_{j \ge 0} = \{L(A \cdot \alpha^j)\}_{j \ge 0}$, очевидно, является бинарной, поскольку L отображает $\mathrm{GF}(2^n)$ и GF(2).

ii) Последовательность $\{s_j\}_{j \ge 0}$ удовлетворяет соотношению (3.2). Чтобы увидеть это, проверим эквивалентное условие (3.3). Линейность L и соотношение $f(\alpha) = \sum_{i=0}^{n} c_i \alpha^i = 0$ влекут

$$\sum_{i=0}^{n} c_i s_{k+i} = \sum_{i=0}^{n} c_i L\left(A \cdot \alpha^{k+i}\right) = L\left(A \cdot \alpha^k \left(\sum_{i=0}^{n} c_i \alpha^i\right)\right) = L(0) = 0.$$

iii) Как мы сейчас покажем, все 2^n выборов $A \in \mathrm{GF}(2^n)$ приводят к различным бинарным решениям соотношения (3.3). По лемме 3.3 они должны составлять все элементы в $\Omega(f)$.

Допустим, что последовательности $\{L(A \cdot \alpha^j)\}_{j\geq 0}$ и $\{L(B \cdot \alpha^j)\}_{j\geq 0}$ идентичны. Из равенств $L(A\cdot\alpha^j) = L(B\cdot\alpha^j), j \geq 0$ и линейности L вытекает, что $L((A-B)\cdot\alpha^j) = 0$ при $0 \leq j < n$. Однако элементы $1, \alpha, \ldots, \alpha^{n-1}$ образуют базис в $\mathrm{GF}(2^n)$, так как f неприводим. Линейность L влечет $L((A-B)\cdot\omega) = 0$ для любого элемента ω из $\mathrm{GF}(2^n)$. Так как отображение L нетривиально, заключаем, что $A = B$.

■

В качестве нетривиального линейного отображения L из $\mathrm{GF}(2^n)$ в $\mathrm{GF}(2)$ удобно брать функцию следа Tr, введенную в задаче B.16. Альтернативой является проекция элемента $\sum_{i=0}^{n-1} a_i\alpha^i$ на его свободный член a_0.

Пример 3.9

Возьмем неприводимый многочлен $f(x) = x^4+x+1$ степени 4 (он даже примитивен), и пусть α — корень $f(x)$, так что $f(\alpha) = 0$. Функция следа задается равенством $Tr(x) = x + x^2 + x^4 + x^8$.

Любой элемент $A \in \mathrm{GF}(2^4) = \left\{\sum_{i=0}^{3} a_i\alpha^i \middle| a_i \in \mathrm{GF}(2), 0 \leq i \leq 3\right\}$ определяет единственную бинарную последовательность $\{s_j\}_{j\geq 0}$, где $s_j = \mathrm{Tr}(A \cdot \alpha^j)$. Ниже мы берем $A = 1 + \alpha + \alpha^2$.

Выходная последовательность, соответствующая произвольному значению A, может быть вычислена с помощью функций PolynomialMod и Table пакета "Mathematica":

```
n = 4; f = 1 + a + a^4; A = 1 + a + a^2;
Tr[x_] := Sum[x^(2^i), {i, 0, n-1}];
s[j_] := PolynomialMod[Tr[A * a^j], {f, 2}];
Table[s[j], {j, 0, 2^n - 2}]
```

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

3.2.5 Криптографические свойства PN-последовательностей

Здесь мы исследуем, в какой степени PN-последовательности удовлетворяют постулатам случайности G1–G3 Голомба. После этого проверим криптографические требования C1–C3. Как всегда, n обозначает длину LFSR.

Ad G1. По лемме 3.2 каждое ненулевое состояние встречается в точности один раз на период. Самый левый бит каждого состояния будет следующим выходным битом. Поэтому число единиц на период равно

2^{n-1}, а число нулей на период равно $2^{n-1}-1$, поскольку нулевое состояние не встречается.

Ad G2. Имеется $2^{n-(k+2)}$ состояний, у которых самые левые $k+2$ компонент имеют вид $0\overbrace{11\ldots1}^{k}0$, соответственно, $1\overbrace{00\ldots0}^{k}1$. Таким образом, блоки и лакуны длины $k \leq n-2$ встречаются в точности $2^{n-(k+2)}$ раз на период.

Состояние $0\overbrace{11\ldots1}^{n-1}$ встречается только один раз. Вслед за ним идет состояние из одних единиц[3], за которым, в свою очередь, идет $\overbrace{11\ldots1}^{n-1}0$. Поэтому имеется один блок длины n и нет блоков длины $n-1$. Аналогично, имеется одна лакуна длины $n-1$ и нет лакун длины n.

Ad G3. Если $\{s_i\}_{i\geq 0} \in \Omega(f)$, то и $\{s_{i+k}\}_{i\geq 0} \in \Omega(f)$ по лемме 3.2. Из линейности $\Omega(f)$ следует тогда, что и $\{s_i + s_{i+k}\}_{i\geq 0} \in \Omega(f)$. Число совпадений на период между $\{s_i\}_{i\geq 0}$ и $\{s_{i+k}\}_{i\geq 0}$ равно числу нулей в одном периоде последовательности $\{s_i + s_{i+k}\}_{i\geq 0}$, которое, как обсуждалось выше, равно $2^{n-1}-1$. Аналогично, число несовпадений равно 2^{n-1}. Поэтому внефазовая автокорреляция $\mathrm{AC}(k)$ равна $-1/(2^n-1)$ для всех $k, 1 \leq k < 2^n - 1$.

Можно утверждать, что постулаты случайности Голомба выполняются для PN-последовательностей наиболее удовлетворительным образом. Проверим теперь C1–C3.

Ad C1. Так как период PN-последовательности, порожденной LFSR длины n, равен 2^n-1, можно легко получать достаточно большие периоды. Например, при $n = 166$ период будет около 10^{50}.

Ad C2. Линейные регистры сдвига крайне просты в реализации.

Ad C3. PN-последовательности весьма небезопасны! В самом деле, знание $2n$ последовательных битов, скажем, $s_k, s_{k+1}, \ldots, s_{k+2n-1}$, позволяет криптоаналитику однозначно определить коэффициенты обратной связи $c_0, c_1, \ldots, c_{n-1}$, а потому — и всю последовательность $\{s_i\}_{i\geq 0}$. Это вытекает из матричного уравнения

$$\begin{pmatrix} s_k & s_{k+1} & \cdots & s_{k+n-1} \\ s_{k+1} & s_{k+2} & \cdots & s_{k+n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{k+n-1} & s_{k+n} & \cdots & s_{k+2n-2} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} s_{k+n} \\ s_{k+n+1} \\ \vdots \\ s_{k+2n-1} \end{pmatrix}. \qquad (3.8)$$

Эта система, как мы сейчас покажем, имеет единственное решение. Если существуют n последовательных линейно зависимых состояний LFSR, т.е. некоторые n последовательных состояний порождают подпространство размерности $\leq n-1$, то — ввиду (3.4) — это же будет верно для любых n последовательных состояний, что, однако, противоречит линейной независимости состояния $(0, 0, \ldots, 0, 1)$ и его $n-1$ последователей.

[3]Поскольку $f(1) \neq 0$.— *Прим. ред.*

Поэтому любые n последовательных состояний (в частности, n строк в матрице выше) линейно независимы. Значит, можно легко определить неизвестные коэффициенты обратной связи $c_0, c_1, \ldots, c_{n-1}$.

Пример 3.10

Допустим, что мы знаем следующую цепочку длины $10: 1,1,0,1,1,1,0,1,0,1$. *Предполагая, что* $n = 5$, *мы можем решить систему* (3.8) *с помощью функции* LinearSolve *из "Mathematica":*

```
m = ( 1 1 0 1 1 )
    ( 1 0 1 1 1 )
    ( 0 1 1 1 0 ) ; b = ( 1 )
    ( 1 1 1 0 1 )       ( 0 )
    ( 1 1 0 1 0 )       ( 1 )
                        ( 0 )
                        ( 1 ) ;
LinearSolve[m, b, Modulus -> 2]
```

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

Коэффициенты обратной связи: $c_0 = 1, c_1 = 0, c_2 = 1, c_3 = 0, c_4 = 0$. *Это легко можно проверить с помощью функций* Table, Mod *и* Do *из пакета "Mathematica":*

```
n = 5;
{c[0], c[1], c[2], c[3], c[4]} = {1, 0, 1, 0, 0};
{s[0], s[1], s[2], s[3], s[4]} = {1, 1, 0, 1, 1};
Do[s[k] = Mod[∑_{i=0}^{n-1} c[i] * s[k-n+i],2], {k, n, 2^n}];
Table[s[k], {k, 0, 2^n - 2}]
```

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

Конечно, в общем случае длина n *используемого LFSR не известна. Мы рассмотрим эту проблему в подразделе* 3.3.1 *в более общей ситуации.*

Если известна лишь цепочка из $2n-1$ последовательных битов PN-последовательности, то коэффициенты обратной связи не обязательно единственны, как ниже показано в примере для $n = 4$ и цепочки 1101011. Это остается верным, даже если мы используем дополнительную информацию о том, что $c_0 = 1$. Ниже мы добавили функцию *NullSpace*, чтобы показать зависимость в линейных соотношениях.

```
m = ( 1 1 0 1 )
    ( 1 0 1 0 ) ; b = ( 0 )
    ( 0 1 0 1 )       ( 1 )
                      ( 1 ) ;
NullSpace[m, Modulus -> 2]
LinearSolve[m, b, Modulus -> 2]
```

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

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

Получаем решения $(1,1,0,0)+\lambda(0,1,0,1)$, где $\lambda \in \{0,1\}$.

Так как последовательности, генерируемые линейными регистрами сдвига, не удовлетворяют требованию С3, следующим шагом должно быть изучение нелинейных регистров сдвига. Однако, поскольку о PN-последовательностях известно уже много, вполне естественно попытаться комбинировать линейные регистры нелинейным способом так, чтобы получать псевдослучайные последовательности с хорошими криптографическими свойствами.

3.3 Нелинейные алгоритмы

3.3.1 Минимальный характеристический многочлен

Как уже упоминалось в начале разд. 3.1, любой детерминированный алгоритм, реализуемый машиной с конечным числом состояний, будет генерировать финально периодическую последовательность $\{s_i\}_{i\geq 0}$, скажем, периода p. Это означает, что $\{s_i\}_{i\geq 0}$, исключая начальную часть, будет порождаться тривиальным образом посредством LFSR с характеристическим многочленом $1+x^p$. Поэтому последовательность, которая, возможно, порождена нелинейным методом, может также порождаться LFSR (если исключить конечную начальную часть). Если эта начальная часть не пуста, не всякое состояние имеет единственного предшественника, и выходная последовательность определенно не будет иметь максимального периода. Мы рассмотрим эту проблему в теореме 3.22. Пока же мы будем предполагать, что выходная последовательность периодична с самого начала. Проведенное обсуждение оправдывает следующее определение.

Определение 3.5
Линейной сложностью (или *линейным эквивалентом*) периодической последовательности называется длина кратчайшего LFSR, порождающего эту последовательность.

Следующие две леммы нужны для доказательства явных утверждений о линейной сложности периодических последовательностей.

Лемма 3.13
Пусть h и f — характеристические многочлены LFSR длин m и n соответственно. Тогда

$$\Omega(h) \subseteq \Omega(f) \Longleftrightarrow h|f.$$

Доказательство. $\Longrightarrow$ Так как $1/h^* \in \Omega(h) \subseteq \Omega(f)$, из следствия 3.5 вытекает, что существует многочлен $u(x)$ степени $< n$, такой, что

$1/h^*(x) = u(x)/f^*(x)$. Отсюда $f^*(x) = h^*(x)u(x)$ и поэтому $f(x) = h(x)u^*(x)$, что означает $h|f$.

$\Longleftarrow$ Запишем $f(x) = a(x)h(x)$, где $\mathrm{degree}(a(x)) = n - m$. Тогда то же следствие 3.5 влечет:

$$\Omega(h) = \left\{ \frac{v(x)}{h^*(x)} \middle| \mathrm{degree}(v(x)) < m \right\} = \left\{ \frac{a^*(x)v(x)}{a^*(x)h^*(x)} \middle| \mathrm{degree}(v(x)) < m \right\} =$$

$$= \left\{ \frac{a^*(x)v(x)}{f^*(x)} \middle| \mathrm{degree}(a^*(x)v(x)) < n \right\} \subseteq \Omega(f).$$

■

Пример 3.11

Последовательность $\{s_i\}_{i\geq 0} = 100101110\ldots$ *является выходной последовательностью LFSR с* $h(x) = 1 + x + x^2$ *и начальным состоянием* $(1, 0, 0)$*; это легко проверить:*

```
n = 3;
{s[0], s[1], s[2]} = {1, 0, 0};
{c[0], c[1], c[2]} = {1, 1, 0};
Do[s[k] = Mod[∑_{i=0}^{n-1} c[i] * s[k-n+i], 2], {k, n, 2^n}];
Table[s[k], {k, 0, 2^n}]
```

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

Однако, поскольку $h(x)(1 + x + x^2) = 1 + x^4 + x^5$*, ту же выходную последовательность можно получить из LFSR с характеристическим многочленом* $f(x) = 1 + x^4 + x^5$ *(см. также пример 3.7). В качестве начального состояния следует взять первые пять членов из* $\{s_i\}_{i\geq 0}$.

```
n = 5;
{s[0], s[1], s[2], s[3], s[4]} = {1, 0, 0, 1, 0};
{c[0], c[1], c[2], c[3], c[4]} = {1, 0, 0, 0, 1};
Do[s[k] = Mod[∑_{i=0}^{n-1} c[i] * s[k-n+i], 2], {k, n, 2^n}];
Table[s[k], {k, 0, 2^n}]
```

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

Пусть $\{s_i\}_{i\geq 0} \in \Omega(f)$ для некоторого f. Предположим, что нужно отыскать многочлен h наименьшей степени, такой,что $\{s_i\}_{i\geq 0} \in \Omega(h)$. Лемма 3.13 предлагает проверить делители f. Следующая лемма говорит нам, когда проверять делители f не нужно.

Лемма 3.14
Пусть $\{s_i\}_{i\geq 0} \in \Omega(f)$ и $S(x) = u(x)/f^*(x)$. Тогда

$$\exists h | f, h \neq f \, [\{s_i\}_{i\geq 0} \in \Omega(h)] \Longleftrightarrow \text{НОД}(u(x), f^*(x)) \neq 1.$$

Доказательство. Пусть $d(x)$ делит НОД$(u(x), f^*(x))$. Тогда

$$S(x) = \frac{u(x)}{f^*(x)} = \frac{u(x)/d(x)}{f^*(x)/d(x)}$$

и поэтому $\{s_i\}_{i\geq 0} \in \Omega(f/d^*)$. Это влечет существование собственного делителя h многочлена f, а именно f/d^*, такого, что $\{s_i\}_{i\geq 0} \in \Omega(h)$.

Доказательство в обратную сторону точно такое же. ■

Теорема 3.15
Пусть $\{s_i\}_{i\geq 0}$ — бинарная периодическая последовательность периода p. Пусть первые p членов последовательности $\{s_i\}_{i\geq 0}$ задаются равенством $S^{(p)}(x) = s_0 + s_1 x + \ldots + s_{p-1} x^{p-1}$. Тогда существует единственный многочлен $m(x)$ со следующими двумя свойствами:

i) $\{s_i\}_{i\geq 0} \in \Omega(m)$,
ii) $\forall h \, [\{s_i\}_{i\geq 0} \in \Omega(h) \Longrightarrow m | h]$.

Взаимный многочлен $m^*(x)$ к $m(x)$ дается равенством

$$m^*(x) = \frac{1 - x^p}{\text{НОД}(S^{(p)}(x), 1 - x^p)}.$$

Многочлен $m(x)$ называется *минимальным характеристическим многочленом* последовательности $\{s_i\}_{i\geq 0}$.

Пример 3.12

Пусть $\{s_i\}_{i\geq 0}$ *имеет период* 15 *и* $S^{(15)}(x) = 1 + x^4 + x^7 + x^8 + x^{10} + x^{12} + x^{13} + x^{14}$. *Тогда*

$$\text{НОД}(x^{15} - 1, S^{(15)}(x)) = (1+x)(1+x+x^2)(1+x+x^2+x^3+x^4)(1+x+x^4).$$

Таким образом, $m^*(x) = (x^{15} - 1)/\text{НОД}(x^{15} - 1, S^{(15)}(x)) = 1 + x^3 + x^4$ *и поэтому* $m(x) = 1 + x + x^4$. *На самом деле данная* $S(x)$ *является выходной последовательностью LFSR на рис.* 3.4.

Эти вычисления можно выполнить с помощью функций PolynomialGCD, PolynomialQuotient и PolynomialMod пакета "Mathematica".

```
p = 15;
S = 1 + x^4 + x^7 + x^8 + x^10 + x^12 + x^13 + x^14;
g = PolynomialGCD[S, x^p - 1, Modulus -> 2];
MSTAR = PolynomialMod[PolynomialQuotient[x^p - 1, g, x], 2]
```

```
1 + x^3 + x^4
```

Доказательство теоремы 3.15. Пусть последовательность $\{s_i\}_{i\geq 0}$ лежит в $\Omega(m)$. Если $\{s_i\}_{i\geq 0} \in \Omega(h)$ для некоторого делителя h многочлена m, то заменим m на h и продолжим эту процедуру, пока не получим $\{s_i\}_{i\geq 0} \notin \Omega(h)$ для любого делителя h многочлена m.

Покажем, что такой многочлен m единственен и имеет вид, указанный в теореме. Так как период $\{s_i\}_{i\geq 0}$ равен p, следствие 3.5 влечет, что для некоторого $u(x)$ с $\text{degree}(u(x)) < \text{degree}(m(x))$

$$\frac{S(p)(x)}{1-x^p} = S^{(p)}(x)(1 + x^p + x^{2p} + \ldots) = S(x) = \frac{u(x)}{m^*(x)}.$$

В силу наших предположений об m и леммы 3.14 $\text{НОД}(m^*(x), u(x)) = 1$, т.е.

$$\text{НОД}(m^*(x), \frac{m^*(x)S^{(p)}(x)}{1-x^p}) = 1.$$

Поэтому

$$\text{НОД}(m^*(x)(1-x^p), m^*(x)S^{(p)}(x)) = 1 - x^p,$$

т.е.

$$m^*(x) \cdot \text{НОД}(1-x^p, S^{(p)}(x)) = 1 - x^p.$$

Следовательно,

$$m^*(x) = \frac{1-x^p}{\text{НОД}(1-x^p, S^{(p)}(x))}.$$

■

Следствие 3.16
Линейная сложность бинарной периодической последовательности $\{s_i\}_{i\geq 0}$ с периодом p и начальным сегментом $S^{(p)}(x) = \sum_{i=0}^{p-1} s_i x^i$ равна

$$p - \text{degree}\left(\text{НОД}(x^p - 1, S^{(p)}(x))\right).$$

3.3.2 Алгоритм Берлекэмпа–Масси

Следствие 3.16 может помочь разработчику нелинейной системы определить, насколько его система безопасна в отношении атаки, описанной в обсуждении "Ad C3" в подразделе 3.2.5.

С другой стороны, криптоаналитик, знающий отрезок выходной последовательности, скажем, $s_0, s_1, \ldots, s_{k-1}$, может испытать следующую стратегию:

i) найти наименьший LFSR, порождающий $s_0, s_1, \ldots, s_{k-1}$;

ii) определить следующий выходной бит этого LFSR с надеждой, что он правильно "предсказывает" следующий бит s_k последовательности.

Определение 3.6
$L_k(\{s_i\}_{i\geq 0})$ — это длина кратчайшего LFSR, порождающего $s_0, s_1, \ldots, s_{k-1}$. Когда из контекста ясно, какова $\{s_i\}_{i\geq 0}$, мы пишем просто L_k. Через $f^{(k)}(x)$ обозначается характеристический многочлен произвольного LFSR длины L_k, порождающего последовательность $s_0, s_1, \ldots, s_{k-1}$.

Ясно, что $L_k(\{s_i\}_{i\geq 0}) \leq k$ для любой последовательности $\{s_i\}_{i\geq 0}$, так как любой LFSR длины k будет порождать $s_0, s_1, \ldots, s_{k-1}$, если в качестве начального состояния взять $(s_0, s_1, \ldots, s_{k-1})$.

Лемма 3.17
Пусть $\{t_i\}_{i\geq 0}$ — выходная последовательность, начинающаяся с $\overbrace{00\ldots0}^{k-1}1$. Тогда $L_k(\{t_i\}_{i\geq 0}) = k$.

Доказательство. Любой LFSR длины $n < k$ с начальным состоянием из первых n символов последовательности $\{t_i\}_{i\geq 0}$ (а все они равны 0) выдаст нулевую последовательность; поэтому t_{k-1} не будет равным 1.

■

Лемма 3.18
Пусть $\{s_i\}_{i\geq 0}$ и $\{t_i\}_{i\geq 0}$ — выходные последовательности. Тогда для всех $k \geq 0$

$$L_k(\{s_i + t_i\}_{i\geq 0}) \leq L_k(\{s_i\}_{i\geq 0}) + L_k(\{t_i\}_{i\geq 0}).$$

Доказательство. Это прямое следствие леммы 3.6. В самом деле, пусть два LFSR с характеристическими многочленами $f^{(k)}(x)$ и $g^{(k)}(x)$ порождают первые k членов последовательностей $\{s_i\}_{i\geq 0}$ и $\{t_i\}_{i\geq 0}$ соответственно. Тогда по лемме 3.6 первые k членов из $\{s_i + t_i\}_{i\geq 0}$ будут порождаться LFSR с характеристическим многочленом равным НОК$[f^{(k)}(x), g^{(k)}(x)]$. Степень последнего не превосходит суммы степеней $f^{(k)}(x)$ и $g^{(k)}(x)$.

■

Из определения 3.6 вытекает, что $L_{k+1} \geq L_k$ для любой последовательности $\{s_i\}_{i\geq 0}$. Но можно сказать больше.

Лемма 3.19
Пусть $\{s_i\}_{i\geq 0}$ — выходная последовательность. Предположим, что LFSR с характеристическим многочленом $f^{(k)}(x)$ не дает правильный выход s_k. Тогда

$$L_{k+1} \geq \max\{L_k, k + 1 - L_k\}.$$

Доказательство. Мы уже знаем, что $L_{k+1} \geq L_k$. Пусть $\{t_i\}_{i\geq 0}$ — последовательность, начинающаяся с $\overbrace{00\ldots0}^{k}1$. Так как LFSR с характеристическим многочленом $f^{(k)}(x)$ порождает $s_0, s_1, \ldots, s_{k-1}$, но не

$s_0, s_1, \ldots, s_k$, этот LFSR будет порождать $\{s_i + t_i\}_{i=0}^{k}$. Поскольку $L_{k+1} \geq$ $\geq L_k$, можно заключить, что

$$L_{k+1}(\{s_i + t_i\}_{i\geq 0}) = L_k(\{s_i + t_i\}_{i\geq 0}) = L_k(\{s_i\}_{i\geq 0})(= L_k).$$

Теперь наше утверждение следует из лемм 3.17 и 3.18 и цепочки

$$k+1 = L_{k+1}(\{t_i\}_{i\geq 0}) \leq L_{k+1}(\{s_i\}_{i\geq 0}) + L_{k+1}(\{s_i + t_i\}_{i\geq 0}) = L_{k+1} + L_k.$$

■

Следующая теорема показывает, что в последней лемме фактически имеет место равенство. Доказательство вытекает из алгоритма Берлекэмпа–Масси, который рекурсивно строит $f^{(k)}(x)$; см. [Mass69]. Этот алгоритм хорошо известен в теории кодирования, где используется при декодировании БЧХ-кодов и кодов Рида–Соломона (см. [Berl68], гл. 7).

Теорема 3.20
Пусть $\{s_i\}_{i\geq 0}$ — выходная последовательность. Предположим, что LFSR с характеристическим многочленом $f^{(k)}(x)$ не дает правильный выход s_k. Тогда

$$L_{k+1} = \max\{L_k, k+1-L_k\}.$$

Доказательство. Ввиду леммы 3.19 достаточно найти многочлен $f(x)$ степени, равной $\max\{L_k, k+1-L_k\}$, который правильно дает первые $k+1$ членов последовательности $\{s_i\}_{i\geq 0}$. Это в точности то, что очень эффективно делает алгоритм Берлекэмпа–Масси.

Мы докажем теорему по индукции.

Начальное индуктивное рассуждение.

Положим $L_0 = 0$ и $f^{(0)}(x) = 1$.

Последовательность $\overbrace{00\ldots0}^{k}$ длины k может порождаться (вырожденным) LFSR с характеристическим многочленом $f^{(k)}(x) = 1$ степени $L_k = 0$.

Последовательность $\overbrace{00\ldots01}^{k}$ длины $k+1$ может порождаться LFSR длины $k+1$, но не более коротким LFSR, как показывает лемма 3.17. В этом случае

$$L_{k+1} = k+1 = k+1-L_k = \max\{L_k, k+1-L_k\},$$

что, в частности, доказывает базу индукции.

Шаг индукции: $k \longrightarrow k+1$.

Заменим в (3.2) k на l. Теперь, положив в (новом) (3.2) $l+n=j$, $c_i = f_i^{(k)}$ и $n = L_k$, индуктивную гипотезу для k можно сформулировать в виде

$$\sum_{i=0}^{L_k-1} f_i^{(k)} s_{j-L_k+i} = s_j, \quad L_k \leq j \leq k-1. \tag{3.9}$$

Если соотношение (3.9) выполняется также при $j = k$, то $L_{k+1} = L_k$, $f^{(k+1)}(x) = f^{(k)}(x)$, и доказывать больше нечего. Если же при $j = k$ равенство (3.9) не выполняется, то

$$\sum_{i=0}^{L_k-1} f_i^{(k)} s_{k-L_k+i} = s_k + 1. \tag{3.10}$$

Пусть m — единственное целое, меньшее k, определяемое условиями
i) $L_m < L_k$,
ii) $L_{m+1} = L_k$,
так что m — индекс последнего приращения L. В силу обоснованного выше начала индукции это число вполне определено. Из индуктивной гипотезы и определения m вытекает, что

$$\sum_{i=0}^{L_m-1} f_i^{(m)} s_{j-L_m+i} = \begin{cases} s_j, & \text{если } L_m \le j \le m-1, \\ s_m + 1, & \text{если } j = m. \end{cases} \tag{3.11}$$

Заметим, что $L_k = L_{m+1} = \max\{L_m, m+1-L_m\} = m+1-L_m$.

Положим $L = \max\{L_k, k+1-L_k\}$. Мы утверждаем, что многочлен

$$\begin{aligned} f(x) &= x^{L-L_k} f^{(k)}(x) + x^{L-(k+1-L_k)} f^{(m)}(x) = \\ &= x^{L-L_k} f^{(k)}(x) + x^{L-k+m-L_m)} f^{(m)}(x) \end{aligned} \tag{3.12}$$

годится в качестве $f^{(k+1)}(x)$. Ясно, что первое слагаемое в (3.12) имеет степень $(L-L_k)+L_k = L$, а второе слагаемое имеет степень $(L-k+m-L_m)+L_m < L$. Поэтому степень многочлена $f(x)$ правильна. Кроме того, в силу соотношений (3.9), (3.10), (3.11)

$$\sum_{i=0}^{L-1} f_i s_{j-L+i} =$$

$$\overset{(3.12)}{=} \sum_{i=L-L_k}^{L-1} f_{i-(L-L_k)}^{(k)} s_{j-L+i} + \sum_{i=L-k+m-L_m}^{L-k+m} f_{i-(L-k+m-L_m)}^{(m)} s_{j-L+i} =$$

$$\overset{\text{замена i}}{=} \sum_{i=0}^{L_k-1} f_i^{(k)} s_{j-L_k+i} + \sum_{i=0}^{L_m-1} f_i^{(m)} s_{j-L_m+m-k+i} + s_{j-k+m} =$$

$$= \begin{cases} s_j + 0 = s_j, & \text{если } L \le j \le k-1, \\ (s_k + 1) + 1 = s_k, & \text{если } j = k. \end{cases}$$

Это доказывает, что LFSR с характеристическим многочленом $f(x)$ действительно порождает последовательность $s_0, s_1, \ldots, s_k$.

■

Теорема 3.20 показывает только, что степень L_k многочлена $f^{(k)}(x)$ единственна. Вообще говоря, сам многочлен $f^{(k)}(x)$ единственным не будет.

Алгоритм, описанный в доказательстве теоремы, может быть формализован следующим образом.

Алгоритм 3.21 (*Берлекэмп–Масси*)

input бинарная последовательность $\{s_i\}_{i\geq 0}$, индекс u;
initialization $f = 1, L = 0, j = 0$;
parameters used
f_{ne}, L_{ne}: характеристический многочлен и длина LFSR, требуемые в данной итерации;
f_{ol}, L_{ol}: многочлен и длина перед последним изменением длины;
diff: разность между номером данной итерации и номером итерации после последнего изменения длины;

while $(s_j = 0) \wedge (j \leq u)$ **do** $j = j + 1$;
if $j = u + 1$ **then** STOP;
put $f_{ol} = 1; L_{ol} = 0$;
$f = x^{j+1}; L = \text{degree}(f)$;
$k = j + 1; \text{diff} = 0$;
while $k < u$ **do**
begin
if $\sum_{i=0}^{L-1} f_i s_{k-L+i} \neq s_k$ **then**
begin
$L_{ne} = \max\{L, k + 1 - L\}$;
$f_{ne} = x^{L_{ne}-L} \cdot f + x^{L_{ne}-(\text{diff}+1-L_{ol})} \cdot f_{ol}$;
if $L_{ne} \neq L$ **then**
begin
$f_{ol} = f; L_{ol} = L$;
$L = L_{ne}; \text{diff} = 0$;
end
else diff = diff + 1;
$f = f_{ne}$;
end
else diff = diff + 1;
$k = k + 1$;
end
output f — характеристический многочлен кратчайшего LFSR, порождающего $(s_0, s_1, \ldots, s_u)$.

Пример 3.13

Рассмотрим последовательность

$\{s_i\}_{i=0}^{30} = \{0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0\}$.

Версия алгоритма Берлекэмпа–Масси, которую мы даем ниже, использует функции Do, CoefficientList, Mod, Max, PolynomialMod, Length и

Print. *Отметим, что мы объединили два предложения "while" алгоритма в одном предложении "do". Все промежуточные многочлены также печатаются.*

```
s = {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0};
Lol = 0; fol = 1;
diff = 0; Clear[x];
f = 1; L = 0; g = CoefficientList[f, {x}];
Do[If[Mod[∑_{i=1}^{L} g[i] s[j - 1 - L + i], 2] == s[j],
    diff = diff + 1;
    Lne = Max[j - L, L];
    fne = PolynomialMod[x^(Lne-L) f + x^(Lne+Lol-diff-1) fol, 2];
If[Lne != L, fol = f; Lol = L; L = Lne; diff = 0;
diff = diff + 1]; f = fne; g = CoefficientList[f, {x}]];
Print["j=", j, ", L=", L, ", f=", f], {j, Length[s]}]
```

```
j=1,   L=0,   f=1
j=2    L=0    f=1
j=3    L=0    f=1
j=4    L=0    f=1
j=5    L=0    f=1
j=6    L=6    f=1 + x^6
j=7    L=6    f=1 + x^5 + x^6
j=8    L=6    f=1 + x^5 + x^6
j=9    L=6    f=1 + x^5 + x^6
j=10   L=6    f=1 + x^5 + x^6
j=11   L=6    f=1 + x^5 + x^6
j=12   L=6    f=x^5 + x^6
j=13   L=6    f=x^5 + x^6
j=14   L=6    f=x^5 + x^6
j=15   L=6    f=x^5 + x^6
j=16   L=6    f=x^5 + x^6
j=17   L=6    f=x^5 + x^6
j=18   L=12   f=1 + x^11 + x^12
j=19   L=12   f=1 + x^10 + x^12
j=20   L=12   f=1 + x^9 + x^12
j=21   L=12   f=1 + x^8 + x^12
j=22   L=12   f=1 + x^7 + x^12
j=23   L=12   f=1 + x^6 + x^12
j=24   L=12   f=1 + x^5 + x^12
j=25   L=13   f=x + x^5 + x^13
```

```
j=1,   L=0,   f=1
j=26   L=13   f=1 + x + x^12 + x^13
j=27   L=14   f=1 + x + x^2 + x^5 + x^12 + x^13 + x^14
j=28   L=14   f=x^2 + x^5 + x^14
j=29   L=14   f=x^2 + x^5 + x^14
j=30   L=16   f=1 + x + x^4 + x^7 + x^12 + x^13 + x^16
j=30   L=16   f=1 + x + x^4 + x^7 + x^12 + x^13 + x^16
```

3.3.3 Несколько замечаний о нелинейных алгоритмах

Общая проблема, связанная с нелинейными регистрами сдвига с обратной связью, состоит в трудности их анализа. Нужно ответить на ряд вопросов: сколько существует различных циклов выходных последовательностей, каковы их длины, какова линейная сложность и т.п. Следующая теорема показывает, что самое малое можно сказать о нелинейных регистрах сдвига с обратной связью.

Ясно, что выходная последовательность нелинейного FSR не будет иметь максимального периода, если существуют два различных состояния с одним и тем же последующим состоянием. Состояние, имеющее более одного предшественника, называется *точкой ветвления*.

Теорема 3.22
Регистр сдвига с обратной связью длины n с (нелинейной) функцией обратной связи $f(s_0, s_1, \ldots, s_{n-1})$ тогда и только тогда не имеет точек ветвления, когда существует такая булева функция $g(s_1, s_2, \ldots, s_{n-1})$, что $f(s_0, s_1, \ldots, s_{n-1}) = s_0 + g(s_1, s_2, \ldots, s_{n-1})$.

Доказательство. Булеву функцию f можно записать в виде

$$f(s_0, s_1, \ldots, s_{n-1}) = g(s_1, s_2, \ldots, s_{n-1}) + s_0 h(s_1, s_2, \ldots, s_{n-1}).$$

$\Longrightarrow$ Если $h(s_1, s_2, \ldots, s_{n-1}) = 0$ для некоторого $(s_1, s_2, \ldots, s_{n-1})$, то состояния $(0, s_1, s_2, \ldots, s_{n-1})$ и $(1, s_1, s_2, \ldots, s_{n-1})$ будут иметь одно и то же последующее состояние. Поэтому, в противоречии с нашим предположением, существует точка ветвления. Следовательно, $h \equiv 1$.

$\Longleftarrow$ Последующим для состояния $(0, s_1, s_2, \ldots, s_{n-1})$ будет $(s_1, s_2, \ldots, s_{n-1}, s_n)$, где $s_n = g(s_1, s_2, \ldots, s_{n-1})$, тогда как для состояния $(1, s_1, s_2, \ldots, s_{n-1})$ последующим будет $(s_1, s_2, \ldots, s_{n-1}, s_n + 1)$. Поэтому точек ветвления нет.

■

Существует много путей использования линейных FSR нелинейным способом. Ниже мы показываем два предложения, которые всесторонне обсуждаются в [Ruep86]. Другие идеи можно найти в [MeOoS97], гл. 6.

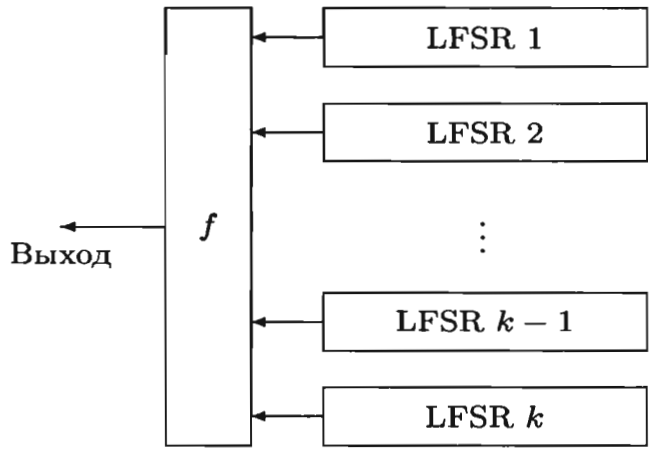


Рис. 3.5. Комбинирование нескольких LFSR с нелинейной функцией f.

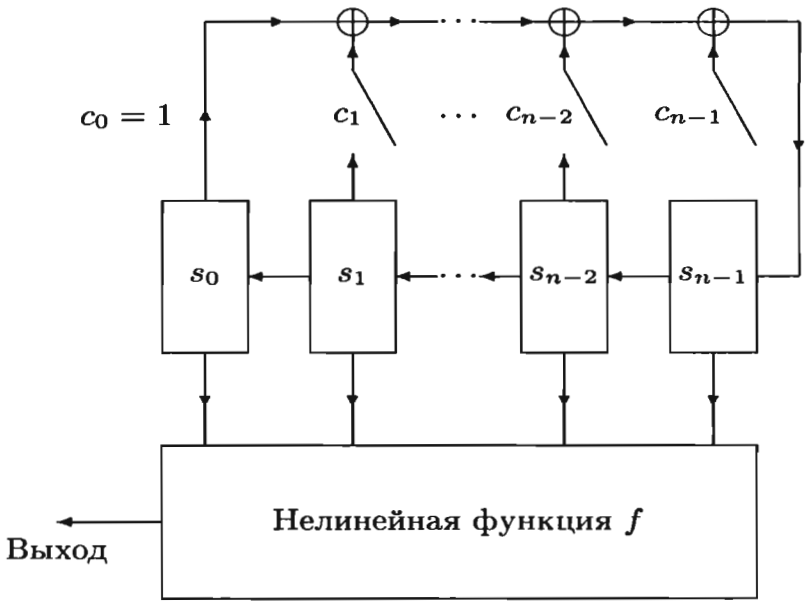


Рис. 3.6. Один LFSR с нелинейным выходом.

3.4 Задачи

Задача 3.1. Пусть $\{s_i\}_{i\geq 0}$ — бинарная периодическая последовательность периода 17 с началом 01101000110001011. В какой степени $\{s_i\}_{i\geq 0}$ удовлетворяет постулатам случайности Голомба? (Замечание для заинтересованного читателя. Указанная последовательность содержит единицы в позициях, соответствующих квадратичным вычетам по модулю 17 — см. строку перед теоремой A.21. Параметры, возникающие при проверке G3, могут быть подсказаны теоремой A.22 и следствием A.24.)

Задача 3.2. Выразите многочлен $\text{НОД}(x^m - 1, x^n - 1)$ в терминах x и $\text{НОД}(m, n)$. (См. также задачу A.3.)

Задача 3.3. Пусть $\{u_i\}_{i\geq 0}$ и $\{v_i\}_{i\geq 0}$ — выходные последовательности

двоичных LFSR длин m и n соответственно, где $m, n \geq 2$. Допустим, что $\{u_i\}_{i\geq 0}$ и $\{v_i\}_{i\geq 0}$ — PN-последовательности и что НОД$(2^m - 1, 2^n - 1) = 1$ (см. задачу А.3). Определим последовательность $\{w_i\}_{i\geq 0}$ равенствами $w_i = u_i v_i, i \geq 0$, и пусть p — период последовательности $\{w_i\}_{i\geq 0}$.

a) Докажите, что p делит $(2^m - 1)(2^n - 1)$.

b) Сколько нулей и сколько единиц содержит подпоследовательность длины $(2^m - 1) \cdot (2^n - 1)$ в $\{w_i\}_{i\geq 0}$?

c) Докажите, что $(2^m - 1)(2^n - 1)/p$ должно делить оба числа, определенные в b).

d) Докажите, что $p = (2^m - 1)(2^n - 1)$.

e) Сколько лакун длины 1 на период имеет последовательность $\{w_i\}_{i\geq 0}$, когда $m, n \geq 4$?

Задача 3.4. Пусть бинарная последовательность $\{s_i\}_{i\geq 0}$ определяется равенствами

$$s_i = \begin{cases} 1, & \text{если } i = 2^l - 1, l \in \mathbb{N}, \\ 0 & \text{в противном случае.} \end{cases}$$

Таким образом, $\{s_i\}_{i\geq 0}$ начинается с 11010001000000010. Пусть L_k — линейная сложность последовательности $s_0, s_1, \ldots, s_{k-1}$. Докажите, что

$$L_{2^l} = 2^{l-1}, \quad l \geq 1.$$

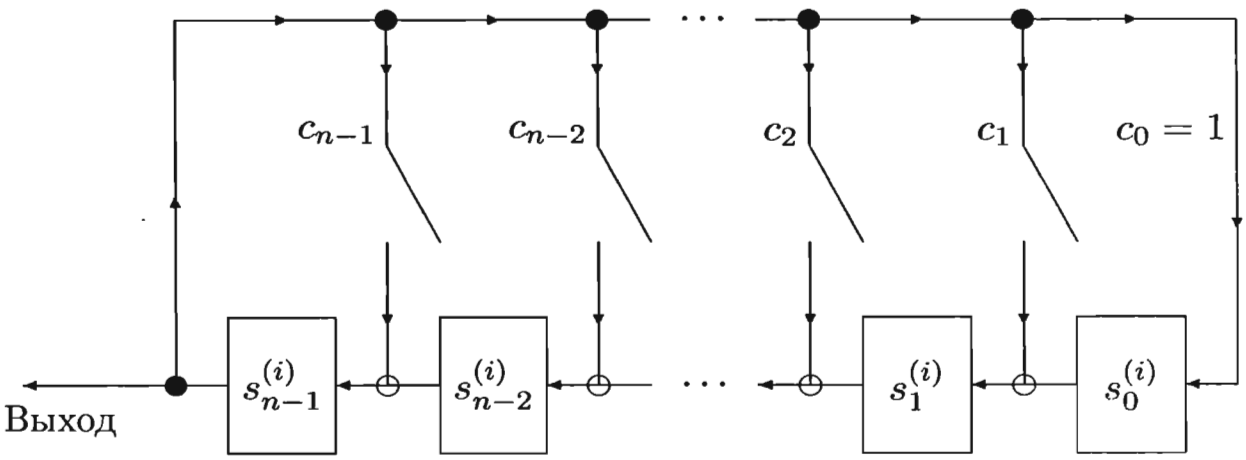
Задача 3.5^M. Пусть бинарная последовательность $\{s_i\}_{i\geq 0}$ имеет период 15 и начинается с 010110000101010. Каков ее минимальный характеристический многочлен? Какова ее линейная сложность?

Задача 3.6. Рассмотрим бинарную периодическую последовательность $\{s_i\}_{i\geq 0}$, определенную периодом $2^{12} - 1$ и значениями $s_0 = s_{2^9-1} = 1$, $s_i = 0$ для $0 < i < 2^{12}, i \neq 2^9 - 1$. Каков ее минимальный характеристический многочлен? Какова ее линейная сложность?

Задача 3.7^M. Рассмотрим бинарные многочлены $f(x) = 1 + x + x^3$ и $g(x) = 1 + x^2 + x^5$. Соответствующие LFSR обозначим через LFSR(f) и LFSR(g). Пусть $\{s_i\}_{i\geq 0}$ и $\{t_i\}_{i\geq 0}$ — выходные последовательности LFSR(f) и LFSR(g) соответственно. Определим последовательность $\{u_i\}_{i\geq 0}$, положив $u_i = s_i + t_i, i \geq 0$.

Различные начальные состояния $(s_0, s_1, s_2, t_0, t_1, t_2, t_3, t_4)$ в количестве 2^8 порождают различные периодические последовательности $\{u_i\}_{i\geq 0}$. Каковы длины циклов (т.е. периоды) этих последовательностей? Дайте начальное состояние каждого цикла.

Задача 3.8. Рассмотрим двоичный регистр сдвига, изображенный ниже.



Пусть $\underline{s}^{(i)} = (s_{n-1}^{(i)}, s_{n-2}^{(i)}, \ldots, s_1^{(i)}, s_0^{(i)})$ — состояние регистра в момент $i \geq 0$.

a) Постройте $n \times n$-матрицу T, удовлетворяющую равенствам $\underline{s}^{(i+1)} = T\underline{s}^{(i)}$ для всех $i \geq 0$.

b) Докажите, что характеристическое уравнение для T над $\mathbb{R}$ имеет вид

$$\lambda^n = c_{n-1}\lambda^{n-1} + c_{n-2}\lambda^{n-2} + \ldots + c_1\lambda + 1.$$

c) Из теории матриц следует, что

$$T^n = c_{n-1}T^{n-1} + c_{n-2}T^{n-2} + \ldots + c_1T + I, \qquad (3.13)$$

где I — единичная $n \times n$-матрица. Так как все элементы в (3.13) целые, равенство (3.13) справедливо также по модулю 2. Выведите рекуррентное соотношение между $\underline{s}^{(i+n)}, \underline{s}^{(i+n-1)}, \ldots, \underline{s}^{(i+1)}$ и $\underline{s}^{(i)}$.

d) Какой LFSR длины n дает ту же выходную последовательность, что и указанный выше регистр сдвига? Каково должно быть начальное состояние в этом LFSR, чтобы генерировать ту же выходную последовательность?

Задача 3.9. Пусть $\alpha \in \mathrm{GF}(2^3)$ — нуль многочлена $f(x) = x^3 + x + 1$. Тогда по теореме B.30

$$\begin{aligned} f(x) &= (x-\alpha)(x-\alpha^2)(x-\alpha^4), \\ f^*(x) &= (x-\alpha^3)(x-\alpha^5)(x-\alpha^6) = (1-\alpha x)(1-\alpha^2 x)(1-\alpha^4 x). \end{aligned}$$

Докажите, что $\Omega(f)$ состоит из всех последовательностей

$$\sum_{i=0}^{\infty}(a\alpha^i + a^2\alpha^{2i} + a^4\alpha^{4i})x^i, a \in \mathrm{GF}(2^3).$$

(Совет: используйте следствие 3.5 и разложение на элементарные дроби над $\mathrm{GF}(2^3)$.) Заметим, что выражение, приведенное выше, можно записать в виде $\sum_{i=0}^{\infty} \mathrm{Tr}(a \cdot \alpha^i)x^i$, где Tr — функция следа, введенная в задаче B.16.

Глава 4

Блочные шифры

4.1 Некоторые общие принципы

4.1.1 Некоторые режимы блочных шифров

□ Режим кодовой книги

Блочные (или *блоковые*) *шифры* представляют собой традиционный вид криптосистем, которые единовременно обрабатывают фиксированное число символов (с помощью данного ключа) и выполняют это шифрование или дешифрование независимо от других входных блоков (см. рис. 4.1). В процессе шифрования данные (открытый текст) поступают в блок шифрования слева, а готовый шифртекст выдается справа. При дешифровании происходит обратный процесс.

В следующем разделе будут описаны некоторые широко используемые блочные шифры. Пока конкретный вид таких шифров не имеет значения. Достаточно представлять их себе как некие электронные устройства, очень быстро преобразующие (с помощью ключа) одну двоичную последовательность длины n в другую так, что обратный процесс возможно выполнить, лишь зная ключ.

Поскольку открытый текст представляет собой длинный двоичный файл, его придется разбить на куски M_i, $i > 0$, длиной по n битов каждый. Результат шифрования куска M_i обозначим через C_i; таким образом, можно записать

$$C_i = BC_k(M_i),\ i \geq 0,$$

где k — это ключ. Процесс дешифрования обозначим через $BC^{\leftarrow}$, тогда $M_i = BC_k^{\leftarrow}(C_i)$.

На последовательности длины n из символов алфавита $\mathcal{A}$ можно смотреть как на отдельные символы алфавита $\mathcal{A}^n$. Разница между n-ками символов одного алфавита и одиночными символами другого, мало значащая с теоретической точки зрения, может иметь большое практическое значение.

Итак, основным свойством блочного шифра является потребность в памяти устройства шифрования.

Ясно, что шифрование двух одинаковых блоков открытого текста при помощи одного и того же ключа даст два одинаковых блока шифртекста. Это очень похоже на использование словаря или шифроблокнота. Из-за

этого режим шифрования, показанный на рис. 4.1, называется *режимом кодовой книги*[1]. Разумеется, шифрование одного блока открытого текста дважды при помощи одного и того же ключа криптогафически небезопасно, поэтому блочные шифры обычно не используются в режиме кодовой книги.

Рис. 4.1. Блочное шифрование в режиме кодовой книги.

□ Режим сцепления блоков

Есть несколько стандартных способов обойти упомянутую выше проблему. Один из них называется *режимом сцепления блоков*. Вновь допустим, что производится шифрование длинного файла. При этом каждый блок шифртекста C_i не только отправляется получателю, но и покоординатно добавляется к следующему блоку открытого текста M_{i+1} перед его шифрованием.

Для выполнения этой процедуры к шифрующему устройству требуется добавить какое-то запоминающее устройство, обычно называемое буфером (см. рис. 4.2). Разумеется, перед началом шифрования буфер должен быть инициализирован.

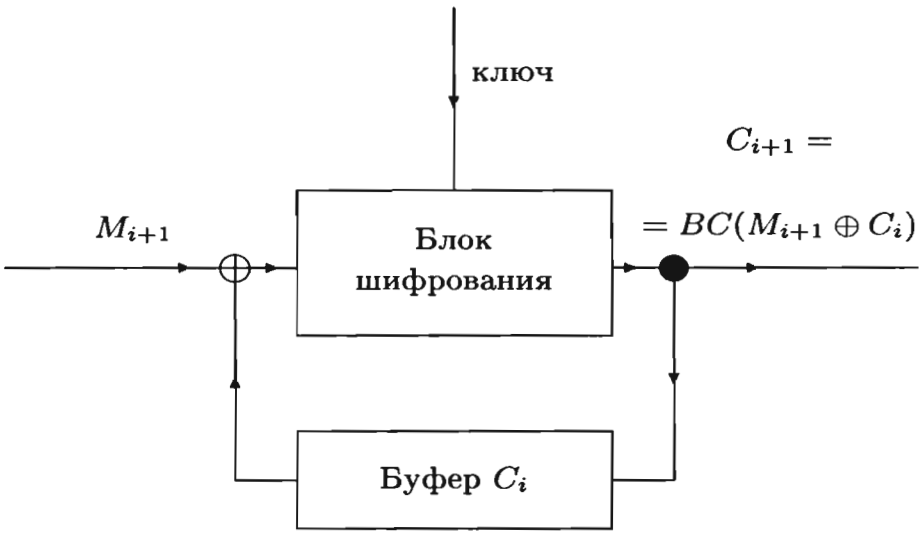


Рис. 4.2. Шифрование в режиме сцепления блоков.

[1] В российском стандарте *[ГОСТ96] этот режим называется *режимом простой замены.— Прим. перев.*

Заметим, что после введения в систему блока памяти возникает устройство, реализующее поточное шифрование.

Процесс дешифрования обратен описанному выше процессу. Буфер должен быть инициализирован точно таким же значением, как и перед процессом шифрования. Это может быть либо часть секретного ключа, либо некоторая фиксированная константа.

Через $BC^{\leftarrow}$ на рис. 4.3 обозначен блок, осуществляющий преобразование, обратное к тому, которое выполняет блок шифрования.

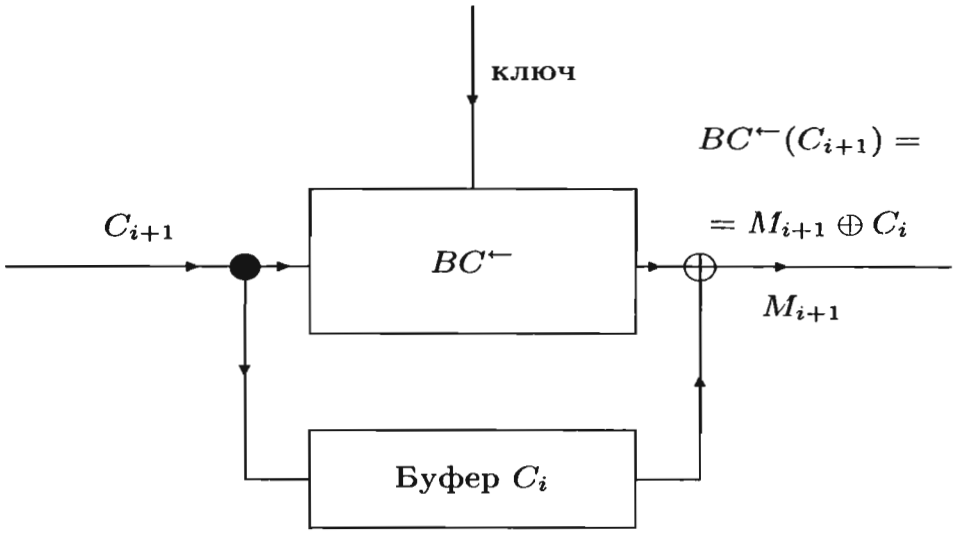


Рис. 4.3. Дешифрование в режиме сцепления блоков.

Замечание. Заметим, что при $C_i = C_j$ для некоторых $i < j$ из рис. 4.2 видно, что $M_j \oplus C_{i-1} = M_j \oplus C_{j-1}$, из чего следует, что $C_{j-1} \oplus C_{i-1} = M_j \oplus M_i$. Т.е. сумма по модулю 2 двух предыдущих блоков шифртекста равна сумме блоков M_i и M_j открытого текста. Во многих ситуациях это может привести к утечке некоторой информации об открытом тексте. В частности, из примера 5.2 можно сделать вывод, что сумма двух английских текстов по модулю 26 (при помощи таблицы Виженера, приведенной в табл. 2.3) сохраняет структуру, достаточную для однозначного восстановления слагаемых.

Это замечание показывает, что имеет смысл выбирать длину блоков большей, чем используемая в настоящее время (т.е. 64 бита)[2].

□ Режим обратной связи с шифртекстом

Другой способ шифрования, позволяющий шифровать одинаковые блоки открытого текста с помощью одного и того же ключа разными блоками шифртекста в разные моменты времени, называется *режим обратной связи с шифртекстом*[3].

[2] В новом стандарте шифрования AES длина блока составляет 128 битов.— *Прим. перев.*

[3] В российском стандарте *[ГОСТ96] аналогом этого режима является *режим гаммирования с обратной связью.— Прим. перев.*

Этод метод изображен ниже на рис. 4.4, но в несколько более обобщенной форме. На практике во многих ситуациях, например, во многих протоколах сети Internet, требуется единовременно пересылать блоки, длина которых (r битов) меньше, чем длина блока шифрования. Вместо чередования r-битовых блоков с $(n-r)$-битовыми последовательностями из нулей и подгонки длины входного блока под блок шифрования можно покоординатно складывать по модулю 2 входной блок с r левыми битами зашифрованного блока. Вход для блока шифрования определяется содержимым регистра сдвига (без обратной связи), которое при каждом такте работы регистра сдвигается на r позиций влево и пополняется справа r битами предыдущего блока шифртекста[4].

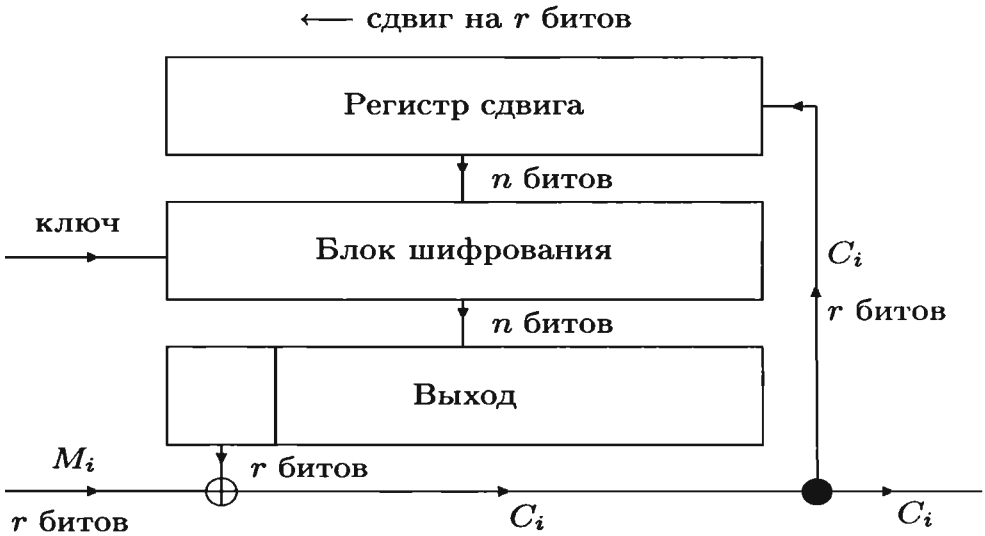


Рис. 4.4. Режим обратной связи с шифртекстом.

4.1.2 Протокол проверки идентичности

В этом подразделе дана идея того, как блочный шифр можно использовать в *протоколе проверки идентичности*. Такой протокол представляет собой общение двух участников, в котором первый желает убедить второго, что в общении участвует именно он, а не кто-то другой. Этот протокол применяется, например, в обмене между устройством чтения карт и smart-картой некоторого человека, скажем Алисы, которая хочет снять деньги со своего счета.

При выдаче smart-карты банк помещает на ней два числа:

- личный номер Алисы Id_A,
- секретный ключ Алисы k_A.

[4]Используются и другие режимы, например, режим обратной связи по открытому тексту, режим генерации MIC (message integrity code — код целостности сообщения; в российском стандарте *[ГОСТ96] "имтитовставка"). Эти режимы (как и предыдущие) детально обсуждаются в *[KaPS95]. Там же во всех подробностях изучаются шифры DES и IDEA, кратко излагаемые ниже.— *Прим. ред.*

Ключ k_A должен быть никому не доступен, даже сама Алиса может его не знать. Личный номер может быть доступен любому устройству чтения карт (его можно даже напечатать на карте снаружи). Эти числа связаны равенством:

$$k_A = \mathrm{BC}_{\mathrm{MK}}(\mathrm{Id}_A), \tag{4.1}$$

где ВС обозначает блочный шифр, а МК — банковкий мастер-ключ. Непрактично хранить в каждом устройстве чтения карт секретные ключи всех клиентов. Вместо этого в каждом устройстве чтения карт хранится МК.

На карте тоже выполняется блочное шифрование ВС.

Когда карта вставлена в устройство чтения, с нее запрашивается личный номер (в нашем случае Id_A). После чего любое подлинное устройство может вычислить секретный ключ Алисы k_A по формуле (4.1).

Далее устройство чтения создает случайную последовательность r из n битов и использует ее как *запрос* (по англ. *challenge*) для карты. Карта возвращает устройству чтения *отклик* (по англ. *response*) $\mathrm{BC}_{k_A}(r)$. Устройству чтения остается лишь проверить это вычисление. Если ответ карты на запрос r правилен, то устройство чтения "узнает", что на карте хранится ключ k_A, и делает вывод об аутентичности карты. В противном случае оно не принимает карту.

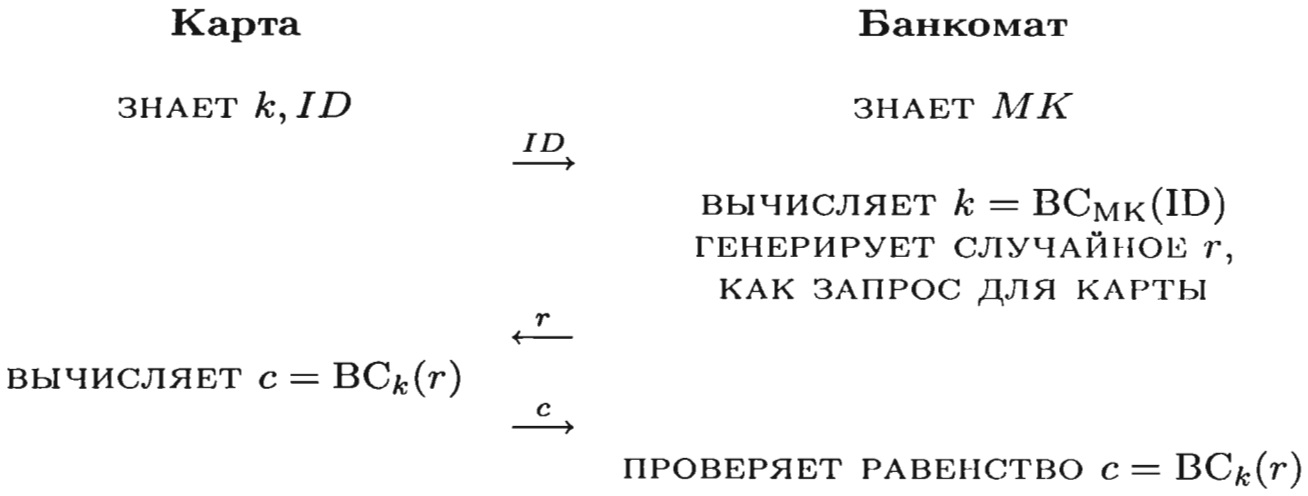


Рис. 4.5. Протокол проверки идентичности.

Карта может использовать тот же протокол для того, чтобы проверить подлинность устройства чтения. Она посылает свой запрос устройству чтения. Отклик устройства может быть корректным только в том случае, когда оно способно вычислить секретный ключ k_A, т.е. когда ему известен мастер-ключ банка МК.

Для связи между картой и ее владельцем обычно используется персональный идентификационный код (PIN-код).

4.2 DES

□ DES

В 1974 г. Национальное бюро стандартов США (NBS) поставило перед промышленностью вопрос о создании криптосистемы, которая могла бы стать стандартом для использования правительством в несекретных приложениях. В ответ на это компанией IBM была создана система LUCIFER. После некоторого видоизменения и упрощения в 1977 г. эта система стала *стандартом шифрования данных* (Data Encryption Standard или, сокращенно, DES).

Сразу же после создания DES были созданы чипы для быстрого аппаратного шифрования. Это сделало очень удобным его применение в больших системах связи. Вся структура DES была полностью опубликована при его введении в качестве стандарта. Никогда ранее такого не делалось, хотя в каждой книге можно найти фразу о том, что безопасность криптосистемы не зависит от того, секретна ли ее структура.

Исчерпывающего описания DES мы не приводим. Его можно найти в [Konh81], [MeyM82], [MeOoV97], [Schn96], *[Сало96] и *[KaPS95].

DES представляет собой блочный шифр с длиной блока 64 бита (см. рис. 4.6).

Ключ состоит из восьми групп по 8 битов. Один бит в каждой из групп — это бит проверки на четность и выбирается так, чтобы количество ненулевых битов в блоке было нечетно. Таким образом, хотя формально длина ключа 64 бита, реально используется 56-битовый ключ.

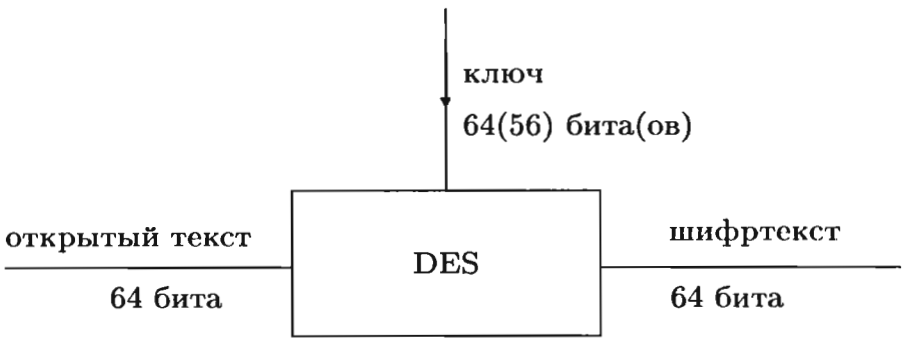


Рис. 4.6. Стандарт шифрования данных (DES).

Работа алгоритма DES состоит из 16 одинаковых раундов. Входной 64-битовый блок разбивается на две части: 32 левых бита составляют L_0, а 32 правых бита составляют R_0.

В каждом раунде новые L и R вычисляются так:

$$\begin{aligned} L_i &= R_{i-1}, & 1 \le i \le 16, \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i), & 1 \le i \le 16, \end{aligned} \tag{4.2}$$

где через K_i обозначен внутренний ключ, т.е. последовательность битов, однозначно определяемая ключом K.

Далее, f — функция от двух аргументов: предшествующей правой половины R_{i-1} и внутреннего ключа K_i, значением которой является последовательность из 32 битов. Эта функция задается с помощью набора фиксированных таблиц, называемых таблицами подстановок. Значение функции покоординатно складывается с L_{i-1}. Левой половиной L_i просто становится предшествующая правая половина. (См. приведенный ниже рис. 4.7.)

Выход DES образуется из L_{16} и R_{16}.

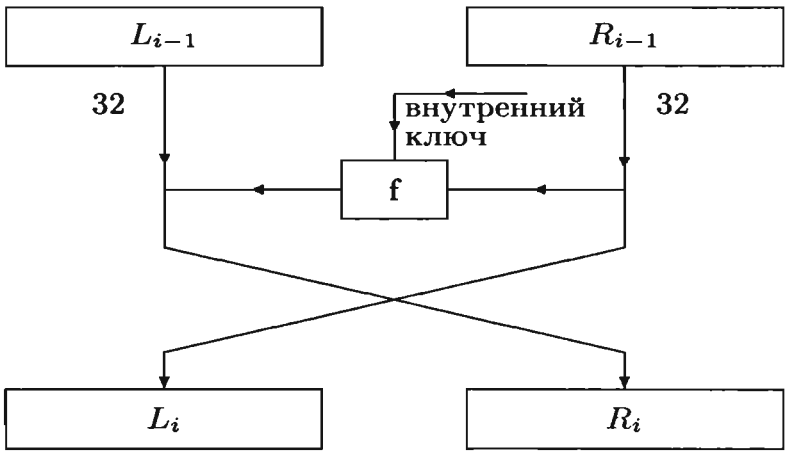


Рис. 4.7. Стандартный раунд DES.

На рис. 4.7 можно увидеть, что процесс, обратный алгоритму DES, происходит по той же схеме, только снизу вверх. Действительно, из формул (4.2) следует, что для всех i, $1 \leq i \leq 16$, выполняются равенства

$$\begin{aligned} R_{i-1} &= L_i, \\ L_{i-1} &= R_i \oplus f(R_{i-1}, K_i) = R_i \oplus f(L_i, K_i). \end{aligned}$$

Многие критично отнеслись к решению сделать DES стандартом. Для этого есть две объективные причины:

i) Эффективная длина ключа (56 битов) слишком мала для организаций с достаточно большими ресурсами. Это делает возможным, по крайней мере в принципе, выполнение исчерпывающего перебора ключей.

ii) Критерии создания таблиц, использующихся в определении функции f, неизвестны. Статистические тесты показывают, что эти таблицы нельзя считать совершенно случайными. Возможно, в их структуре скрыта какая-либо секретная лазейка.

В течение первых двадцати лет после публикации алгоритма DES не появлялось никаких сообщений о способах его эффективного взлома.

Однако, в 1998 году DES впервые был взломан с помощью более или менее лобовой атаки[5].

□ Тройной DES

Когда стало ясно, ято DES более не пригоден для защиты конфиденциальной информации, появилась его модификация, называемая *тройной DES*. Он состоит из трех последовательных выполнений алгоритма DES, с той лишь особенностью, что на втором шаге выполняется обратный алгоритм. Т.е. сначала DES, затем $DES^{\leftarrow}$, и наконец снова DES. (См. приведенный ниже рис. 4.8.)

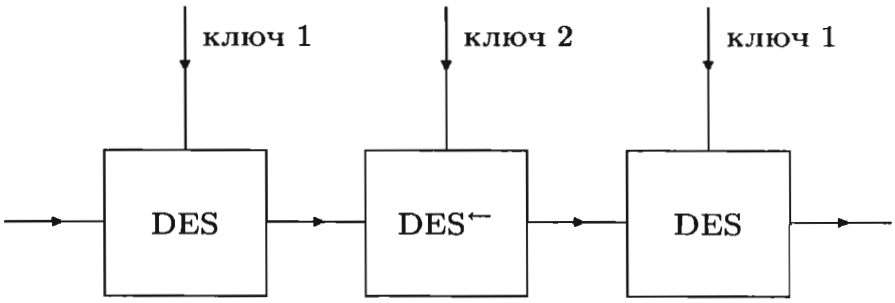


Рис. 4.8. Тройной DES.

Отметим пару особенностей этой схемы. Во-первых, третий ключ совпадает с первым. Тем самым реальная длина ключа составляет $2 \times 56 = 112$ битов. Считается, что это обеспечит безопасность системы на несколько лет.

Во-вторых, на среднем шаге вместо DES применяется $DES^{\leftarrow}$.

Две эти особенности позволяют сделать системы, в которых используется тройной DES, совместимыми с системами, в которых используется обычный DES. В самом деле, взяв одинаковые ключи 1 и 2, можно превратить описанную выше систему в обычный DES.

4.3 IDEA

Имеется и несколько альтернатив алгоритму DES. Их рассмотрение представляет интерес из-за экспортных ограничений, накладываемых американским правительством, а также из-за высокой стоимости патентных прав. В отличие от DES, использующего в каждом своем раунде фиксированные таблицы, в некоторых его альтернативах применяются элементарные математические операции.

Одной из таких систем является IDEA [Lai92]. Ее название — это аббревиатура от International Data Encryption Algorithm (международный

[5]Об атаках на DES см. интересный обзор *[Land00]. Российский симметричный криптографический алгоритм *[ГОСТ96] построен на тех же принципах, что и DES, но гораздо надежнее последнего, так как имеет секретный ключ длиной 256 битов (плюс 512 битов долговременного ключа).— *Прим. ред.*

алгоритм шифрования данных). Длина блока в IDEA также составляет 64 бита (см. замечание в подразд. 4.1.1 об этой величине), но длина ключа — уже 128 битов. Работа алгоритма состоит из 8 одинаковых раундов, изображенных ниже на рис. 4.9. Последовательность из 64 битов разбивается на четыре блока по 16 битов каждый. Эти блоки на входе стандартного раунда обозначены через X_i, $1 \le i \le 4$, а на выходе — через Y_i, $1 \le i \le 4$. Через K_i, $1 \le i \le 4$, обозначены подстроки ключевой строки. Их конкретное расположение зависит от номера раунда.

Элементарные математические операции в IDEA выполняются на этих 16 битах. Опишем эти операции.

- **Покоординатное XOR** (сложение по модулю 2)

На рис. 4.9 эта операция изображается символом $\oplus$.

В пакете "Mathematica" операция XOR может быть выполнена с помощью функции *Mod* (здесь показано, как она работает на четверках).

```
Mod[{1, 1, 0, 0} + {1, 0, 1, 0}, 2]
```

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

- **Сложение по модулю 2^{16}**

На рис. 4.9 эта операция изображается в виде квадратика со знаком плюс внутри: $\boxplus$.

Для выполнения этой операции будем считать входные 16-битовые блоки двоичными представлениями двух целых чисел. Тогда выходным блоком будет двоичное представление суммы этих чисел по модулю 2^{16}.

В пакете "Mathematica" это можно выполнить с помощью функций *FromDigits*, *IntegerDigits* (здесь показано, как они работают на четверках).

```
a = FromDigits[{1, 0, 1, 1}, 2]
b = FromDigits[{1, 1, 1, 0}, 2]
su = Mod[a + b, 16]
IntegerDigits[su, 2]
```

```
11
```

```
14
```

```
9
```

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

- **Умножение по модулю $2^{16} + 1$**

На рис. 4.9 эта операция изображается символом $\otimes$.

Для выполнения умножения будем считать входные 16-битовые блоки двоичными представлениями двух целых чисел. Все эти числа, за

исключением нуля, обратимы по модулю простого числа $2^{16}+1 = 65537$. Отождествим блок из 16 нулей не с числом 0, а с числом 2^{16}, тогда получим взаимно-однозначное соответствие между 16-битовыми блоками и элементами мультипликативной группы $\mathbb{Z}^*_{65537}$ (см. пример В.3). Выходным блоком будет двоичное представление произведения этих чисел по модулю $2^{16}+1$ (исключая $1\overbrace{0...0}^{16}$, которое следует заменить на $\overbrace{0...0}^{16}$).

Поскольку число $2^{16}+1$ простое, определенное выше умножение $a \otimes b$ будет взаимно-однозначным отображением при фиксированном a или b. Ниже приведен пример выполнения этой операции для четверок. Заметим, что число 2^4+1 — тоже простое.

```
a = FromDigits[{1, 0, 1, 0}, 2];
b = FromDigits[{0, 1, 1, 0}, 2];
a = If[a == 0, 16, a];
b = If[b == 0, 16, b];
pr = Mod[a * b, 17]
pr = If[pr == 0, 16, pr];
IntegerDigits[pr, 2, 4]
```

```
9
```

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

Читатель может попробовать перемножить последовательности $\{1,0,0,0\}$ и $\{0,0,1,0\}$.

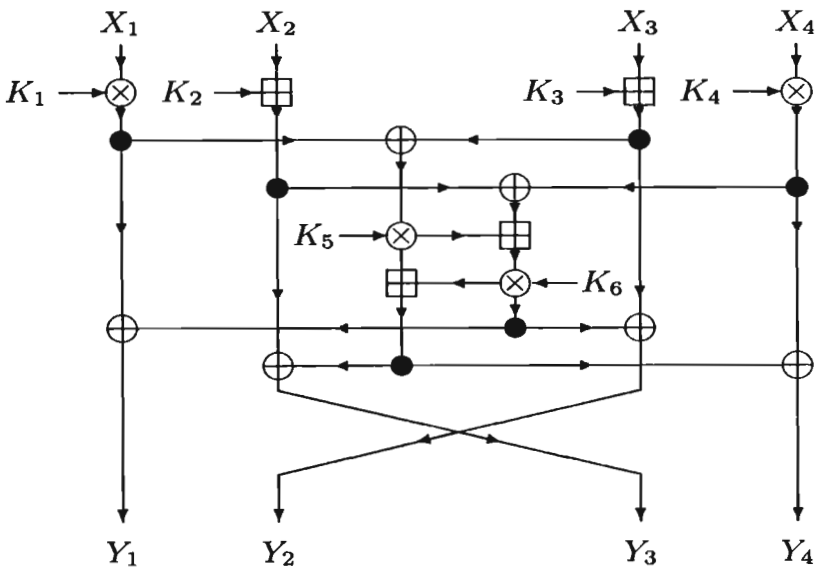


Рис. 4.9. Один раунд международного алгоритма шифрования данных (IDEA).

Обратный алгоритм для IDEA, как и для DES, получается при движении по схеме снизу вверх.

4.4 Дополнительные замечания

Алгоритм *RC5* (см. *[Молд98]) работает по схеме, напоминающей IDEA. Одной из элементарных операций в этой системе вновь является сложение по модулю 2^w, где w — это длина слова, а вместо умножения по модулю $2^w + 1$, работающего лишь при простом $2^w + 1$, в RC5 применяются циклические сдвиги.

Длина блока в RC5 равна $2w$, где w можно вибирать из трех возможных значений: 16, 32 или 64. Дополнительным преимуществом RC5 является свобода выбора количества раундов в схеме. В зависимости от требуемой скорости работы и безопасности пользователь может зафиксировать большее или меньшее число раундов.

В 1993г. были опубликованы два способа атаки на блочные шифры, оказавшихся неожиданно сильными. Эти методы называются *линейным* и *дифференциальным криптоанализом* (см. [MatsY93] и [BihS93], соответственно[6]) и представляют собой разновидности атак с известным открытым текстом. Некоторые из предложенных блочных шифров оказались слабо защищены от подобных атак, но алгоритм DES выдерживает их. Позже стало ясно, что разработчики DES были осведомлены о подобных способах атак. Для получения более подробных сведений можно обратиться к [Knud94].

Во время написания этой книги американский национальный институт стандартов и технологии (сокращенно NIST) исследовал несколько проектов криптосистем с целью выбора нового промышленного стандарта. Названия этих криптосистем — CAST-256, CRYPTON, DFC, DEAL, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, RIJNDAEL, SAFER+, SERPENT и TWOFISH (см. веб-сайт 'Advanced Encryption Standard' http://csrc.nist.gov/encryption/aes/aes_home.htm). Результат этого исследования пока неясен[7].

4.5 Задачи

Задача 4.1. Опишите процесс дешифрования в режиме обратной связи по шифртексту.

Задача 4.2. Рассмотрим блочный шифр, используемый в режиме сцепления блоков. Предположим, что в процессе передачи i-й блок C_i шифртекста оказался испорчен. На какое число блоков открытого текста это повлияет? Ответьте на тот же вопрос для режима обратной связи с шифртекстом.

Задача 4.3^M. Можно ли в системе IDEA, сохранив общую схему и элементарные операции, увеличить длину рабочих блоков? (Длина таких блоков в IDEA составляет 16 битов.) Какая проблема при этом возникает?

---

[6] На русском языке эти методы (наряду с другими) изложены в *[РосМ01]. *Прим. ред.*

[7] Теперь уже ясен. Летом 2001г. NIST в качестве стандарта AES утвердил проект RIJNDAEL. Его детальное описание см., например, в *[DaRi99].— *Прим. ред.*

Глава 5

Теория Шеннона

5.1 Энтропия, избыточность и расстояние единственности

В главе 2 мы видели, что криптоанализ криптосистемы часто зависит от имеющейся в тексте структуры. Например, в табл. 2.1 мы отыскали ключ 22 (или -4) из-за того, что открытый текст "tu quoque Brute" оказался единственным осмысленным из возможных.

Структура открытого текста сохраняется и в шифртексте, хотя и в скрытой форме. Если информация, содержащаяся в этой структуре, превышает неопределенность ключа, то может появится возможность найти открытый текст по криптограмме!

Сначала определим понятие информации. Пусть X — случайная величина, распределение которой на множестве $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ задается вероятностями

$$\Pr_{\mathcal{X}}(X = x_i) = p_i, \quad 1 \leq i \leq n.$$

Разумеется, $\sum_{i=1}^{n} p_i = 1$ и $p_i \geq 0$ для всех $1 \leq i \leq n$. Покажем, что величина

$$J(p_i) = -\log_2 p_i \tag{5.1}$$

хорошо подходит для измерения количества *информации*, содержащейся в сообщении о том, что произойдет событие x_i, $1 \leq i \leq n$. В основании логарифма число 2 можно заменить на любое другое, но, как будет видно ниже, двойка лучше всего отражает наше интуитивное понимание информации. В случае, когда в основании логарифма стоит 2, единица информации называется *бит*.

Пусть $\mathcal{X} = \{x\}$, т.е. $n = 1$; тогда $p_1 = 1$. Теперь событие x достоверно, и сообщение о том, что оно произойдет, не несет никакой информации (как, например, фраза "Солнце завтра снова взойдет"). Это прекрасно соответсвует формуле (5.1), поскольку $J(1) = 0$.

Теперь рассмотрим событие, происходящее с вероятностью 1/2, типа рождения ребенка определенного пола. Тогда $\mathcal{X} = \{\text{м}, \text{д}\}$. В предположении, что дети обоих полов рождаются с одинаковой вероятностью, ясно, что сообщение о поле новорожденного дает ровно один бит информации. Например, 1 означает рождение мальчика, а 0 — рождение девочки. Это вновь согласуется с формулой (5.1), поскольку $J(1/2) = 1$.

Если некоторое событие происходит с вероятностью $1/4$, то сообщение о том, что оно происходит, дает два бита информации. Это очевидно в ситуации с четырьмя равновозможными исходами. Каждому из этих исходов можно сопоставить свою двоичную последовательность длины два. С другой стороны, количество информации, которую несет сообщение о событии, происходящем с вероятностью $1/4$, не должно зависеть от вероятностей других возможных событий. Таким образом, значение $J(1/4) = 2$, вычисленное по формуле (5.1), вновь согласуется с нашей интуицией. Продолжая подобным образом, получаем, что

$$J(1/2^k) = k, \ k \geq 0. \tag{5.2}$$

Математическое ожидание случайной величины $J(\Pr_{\mathcal{X}}(X))$, определенной на множестве $\mathcal{X}$, называется *энтропией* X и обозначается либо через $H(X)$, либо через $H(\underline{p})$, где $\underline{p} = (p_1, p_2, ..., p_n)$. Следовательно,

$$H(X) = M(J(\Pr_{\mathcal{X}}(X))) = \sum_{i=1}^{n} p_i J(p_i) = -\sum_{i=1}^{n} p_i \log_2 p_i;$$

$$H(\underline{p}) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{5.3}$$

При $n = 2$ принято обозначать p_1 через p, а p_2 через q и писать $h(p)$ вместо $H(\underline{p})$.

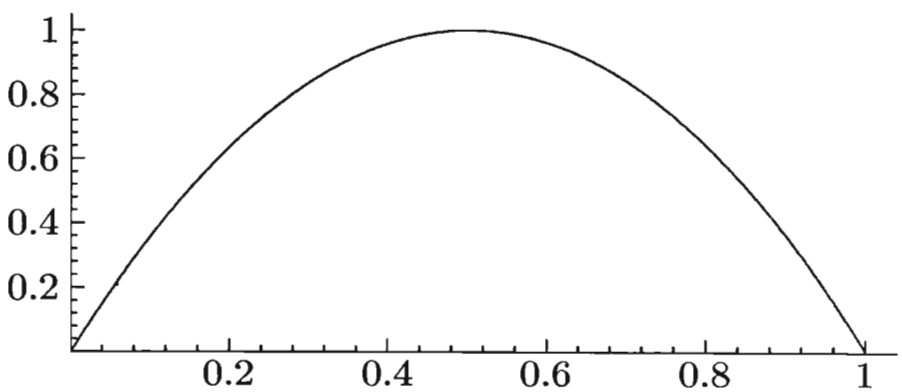
$$h(p) = -p \cdot \log_2 p - (1-p) \cdot \log_2(1-p), \ 0 \leq p \leq 1. \tag{5.4}$$

Поскольку $x \cdot \log_2 x$ стремится к 0 при $x \longrightarrow 0$, не возникает проблем с определением функции энтропии $H(\underline{p})$ при равенстве нулю (или 1) некоторых из вероятностей p_i.

График функции $h(p)$ изображен ниже при помощи функции *Plot* пакета "Mathematica".

```
p=.;
Entropy[p_] = -p * Log[2, p] - (1 - p) * Log[2, 1 - p];
```

```
Plot[Entropy[x], {x, 0, 1}]
```



Функцию энтропии $H(\underline{p})$ можно вычислить следующим образом.

```
MultiEntropy[p_List] :=- Sum_{i=1}^{Length[p]} p[[i]] * Log[2, p[[i]]]
```

```
p={1 / 4, 1 / 4, 1 / 4, 1 / 4};
MultiEntropy[p]
```

```
2
```

Можно дать следующие интерпретации энтропии $H(X)$ случайной величины X.

- среднее количество информации, которое дает сообщение о значении X,
- неопределенность X,
- среднее количество битов, небходимое для реализации X.

Помня эти интерпретации, можно ожидать, что функция энтропии $H(X)$ обладает следующими свойствами.

P1: $H(p_1, p_2, ..., p_n) = H(p_1, p_2, ..., p_n, 0)$.

P2: $H(p_1, p_2, ..., p_n) = H(p_{\sigma(1)}, p_{\sigma(2)}, ..., p_{\sigma(n)})$ для любой перестановки σ множества $\{1, 2, ..., n\}$.

P3: $0 \leq H(p_1, p_2, ..., p_n) \leq H(1/n, 1/n, ..., 1/n)$.

P4: $H(p_1, p_2, ..., p_n) = H(p_1, p_2, ..., p_{n-2}, p_{n-1} + p_n) +$
$+(p_{n-1} + p_n) H\left(\frac{p_{n-1}}{p_{n-1}+p_n}, \frac{p_n}{p_{n-1}+p_n}\right)$.

Данные свойства можно интерпретировать следующим образом.

P1 говорит, что добавление к множеству $\mathcal{X}$ невозможного события не влияет на неопределенность величины X.

P2 утверждает, что перенумерование различных событий множества $\mathcal{X}$ оставляет энтропию неизменной.

P3 говорит, что неопределенность величины X максимальна, если все события равновероятны.

И, наконец, P4 утверждает, что среднее количество двоичных разрядов, небходимое для описания результата из множества $\mathcal{X}$ равно числу битов, необходимых при объединении событий x_{n-1} и x_n в одно событие $\hat{x}_{n-1}$, плюс число битов, необходимых для того, чтобы различить события x_{n-1} и x_n при условии, что событие $\hat{x}_{n-1}$ произошло.

Например, если $n = 4$, то $H(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) = 2$ и

$$H(\tfrac{1}{4},\tfrac{1}{4},\tfrac{1}{2}) + \tfrac{1}{2}\cdot H(\tfrac{1}{2},\tfrac{1}{2}) = (\tfrac{1}{2}\cdot 2 + \tfrac{1}{4}\cdot 2 + \tfrac{1}{2}\cdot 1) + \tfrac{1}{2}\cdot 1 = 2.$$

Хотя доказательства мы здесь не приводим, можно показать (см. [Khin57]), что функция, определяемая формулой (5.1), — единственная непрерывная функция, удовлетворяющая условию (5.2) и порождающая функцию энтропии $\sum_{i=1}^{n} p_i J(p_i)$, обладающую приведенными выше свойствами P1—P4.

Пример 5.1

Рассмотрим подбрасывание монеты. Пусть $Pr(\text{орел}) = p$ *и* $Pr(\text{решка}) = 1 - p$, $0 \le p \le 1$. *Энтропия задается формулой* (5.4).

Разумеется, условие $h(1/2) = 1$ *подтверждается тем фактом, что для представления результата такого эксперимента с правильной монетой необходим один бит. Например,* $0 \leftrightarrow$ *"орел", а* $1 \leftrightarrow$ *"решка".*

Поскольку $h(1/4) \approx 0.8113$, *можно ожидать, что в среднем для представления результата эксперимента с неправильной монетой, для которой* $Pr(\text{орел}) = 1/4$, *окажется достаточно* 0.8113 *бита. Это утверждение верно в том смысле, что можно сколь угодно приблизиться к числу* 0.8113. *В гл.* 6 *будет показано, как это сделать. Трюк состоит в том, что результат большого числа бросаний представляется единственной двоичной строкой. Например, результат двух бросаний можно представить следующим образом:*

выпало	*вероятность*	*представление*
оо	1/16	111
ор	3/16	110
ро	3/16	10
рр	9/16	0

Математическое ожидание длины такого представления равно

$$\frac{1}{16}\cdot 3 + \frac{3}{16}\cdot 3 + \frac{3}{16}\cdot 2 + \frac{9}{16}\cdot 1 = \frac{27}{16}.$$

Но при этом описаны результаты сразу двух бросаний, т.е. при такой схеме на одно бросание приходится по $27/32 \approx 0.843$ *бита. Объединяя в одну группу по три, четыре и более бросаний, будем получать все более и более точное приближение для величины* $h(1/4)$.

Существует, однако, проблема однозначного декодирования, состоящая в том, как получатель по длинной строке из нулей и единиц может определить, сколько экспериментов проводилось и какой был результат у каждого из них. Нетрудно проверить, что любая последовательность, состоящая из подпоследовательностей 111, 110, 10 *и* 0, *однозначно разбивается на такие блоки. Мы займемся этой проблемой более подробно в гл.* 6.

Пример 5.2 (часть 1)

При кодировании достаточно длинных последовательностей из 26 *букв английского алфавита двоичными последовательностями можно добиться того, что на каждую букву будет приходиться по* $\log_2 26 \approx 4.70$ *бита. Действительно, для кодирования* k*-буквенных последовательностей потребуется* $\lceil \log_2 26^k \rceil$ *битов, т.е. на каждую букву будет приходиться по* $\lceil \log_2 26^k \rceil / k$ *битов, а эта последовательность сходится к* $\log_2 26$ *при* $k \longrightarrow \infty$.

С другой стороны, энтропию 1*-грамм легко вычислить по вероятностям, приведенным в табл.* 1.1. *Получится* 4.15 *бита на букву.*

Подобные вычисления были проведены для биграмм и триграмм (см. [МеуМ82], приложение F). Получены следующие величины:

*H(*1*-грамм)*	≈ 4.15 *бита на букву*
*H(*2*-грамм)/*2	≈ 3.62 *бита на букву*
*H(*3*-грамм)/*3	≈ 3.22 *бита на букву*

*По некоторым тестам асимптотическое значение величины H(*n*-грамм)/*n *при* $n \longrightarrow \infty$ *меньше, чем* 1.5 *бита на букву!*

Определение 5.1
Обозначим через $(X_0,\ X_1,\ ..., X_{n-1})$, $n \geq 1$, открытый текст, порожденный источником открытых текстов $\mathcal{S}$ над алфавитом $\mathbb{Z}_2$. Тогда *избыточность* D_n текста $(X_0,\ X_1,\ ..., X_{n-1})$ определяется равенством

$$D_n = n - H(X_0,\ X_1,\ ..., X_{n-1}).$$

Число $\delta = D_n/n$ является мерой избыточности, приходящейся на одну букву текста.

Если алфавит состоит из q букв, то каждый символ представляется с помощью $\log_2 q$ битов, а избыточность задается равенством $D_n = n \cdot \log_2 q - H(X_0,\ X_1,\ ..., X_{n-1})$. Если допустить использование представлений разной длины для разных символов алфавита со средней длиной представления ℓ битов на символ, то получим $D_n = n \cdot \ell - H(X_0,\ X_1,\ ..., X_{n-1})$.

Избыточность является мерой того, насколько длина данного текста превосходит длину текста, минимально необходимого для хранения содержащейся в данном тексте информации (все измеряется в битах).

Теперь обратимся к криптосистеме $\mathcal{E}$, состоящей из криптографических преобразований E_k, проиндексированных ключами k из ключевого пространства $\mathcal{K}$. Предположим, что неизвестный открытый текст представляет собой правильный текст на английском языке. Тогда, имея на руках шифртекст, криптоаналитик может попытаться, перебирая все возможные ключи, получить все варианты открытых текстов. Поскольку шифртекст состоит из нескольких букв, некоторые ключи придется

признать недопустимыми из-за того, что они приводят к невозможным или маловероятным комбинациям букв в открытом тексте. При этом чем длиннее будет шифртекст, тем больше ключей окажутся отвергнутыми из-за того, что они разрушают структуру или смысл английского текста. Говоря более формально, они разрушают избыточность открытого текста. Рано или поздно останется единственный возможный кандидат на роль ключа шифрования.

Вернемся к обобщенной постановке. Пусть длина открытого текста равна n битов. Имеется 2^n двоичных последовательностей, но лишь $2^{H(X_0,\ X_1,\ ...,X_{n-1})}$ из них представляют собой осмысленные сообщения. Вероятность того, что дешифрование при помощи ошибочного ключа приведет к допустимому сообщению, равна $2^{H(X_0,\ X_1,\ ...,X_{n-1})}/2^n$. Если перепробовать все возможные ключи, то можно ожидать, что получится $|\mathcal{K}|\cdot 2^{H(X_0,\ X_1,\ ...,X_{n-1})}/2^n$ осмысленных сообщений. Причем если все ключи равновероятны, а K — случайная величина, задающая распределение ключей в пространстве $\mathcal{K}$, то $|\mathcal{K}| = 2^{H(K)}$. Тогда число осмысленных сообщений равно $2^{H(K)+H(X_0,\ X_1,\ ...,X_{n-1})}/2^n$. Если это число не превосходит 1, то весьма вероятно, что лишь один ключ приводит к осмысленному дешифрованию данного шифртекста. Это происходит при

$$H(K) + H(X_0,\ X_1,\ ...,X_{n-1}) - n \leq 0,$$

т.е. в ситуации, когда избыточность удовлетворяет условию

$$D_n \geq H(K).$$

Если величина K не является равномерно распределенной, то при обосновании полученной выше оценки мы все равно будем считать величину $H(K)$ мерой неопределенности ключа.

Определение 5.2
Пусть в криптосистеме $\mathcal{E}$ используются ключи из ключевого пространства $\mathcal{K}$, а открытые тексты порождаются источником $\mathcal{S}$. Рассмотрим атаку против криптосистемы $\mathcal{E}$ на основе известного шифртекста. *Расстояние единственности* этой криптосистемы определяется как

$$\min\{n \in \mathbb{N}_+ | D_n \geq H(K)\},$$

где $H(K)$ — энтропия ключа, а D_n — избыточность открытого текста.

Смысл такого названия в том, что если избыточность открытого текста превосходит неопределенность ключа, то криптоаналитик, обладающий достаточными вычислительными ресурсами, может восстановить открытый текст по криптограмме. Таким образом, расстояние единственности показывает пользователю криптосистемы, когда необходимо сменить ключ для того, чтобы обеспечить безопасность системы.

Пример 5.2 (часть 2)

Продолжим обсуждение примера 5.2. *Предположим, что к английскому тексту применяется простая подстановка (см. подразд.* 2.1.2*). Предположив, что все* 26! *ключей равновероятны, находим*

$$H(K) = -\sum_{i=1}^{26!} \frac{1}{26!} \log_2 \frac{1}{26!} = \log_2 26! \approx 88.382 \text{ бита.}$$

Оценив приблизительно избыточность D_n *английского текста из* n *букв числом* $(4.70-1.50)n = 3.20n$ *бита, получим, что расстояние единственности равно* $88.4/3.2 \approx 28$ *букв.*

Процитируем Фридмана [Frie73]: "практически каждый пример из 25 *или более букв, представляющий собой результат действия одноалфавитной подстановки на осмысленный английский текст, может быть легко расшифрован". Отметим, что числа* 25 *и* 28 *прекрасно согласуются.*

5.2 Взаимная информация и безусловно безопасные системы

Часто одна случайная величина содержит информацию о другой. В криптосистемах открытый текст и шифртекст связаны посредством ключа. В данном разделе мы дадим формальное определение (в теоретико-информационном смысле этого слова) абсолютно безопасной криптосистемы.

Пусть X и Y — случайные величины, определенные на множествах $\mathcal{X}$ и $\mathcal{Y}$ соответственно. Вероятности $\Pr_{\mathcal{X},\mathcal{Y}}(X = x, Y = y)$ или сокращенно

$$p_{\mathcal{X},\mathcal{Y}}(x, y)$$

задают *совместное распределение* величин X и Y. Вероятность того, что $X = x$ при условии, что $Y = y$, называется *условной вероятностью* и обозначается $\Pr_{\mathcal{X}|\mathcal{Y}}(X = x|Y = y)$ или

$$p_{\mathcal{X}|\mathcal{Y}}(x|y).$$

Выполняется соотношение

$$p_{\mathcal{X},\mathcal{Y}}(x, y) = p_{\mathcal{X}|\mathcal{Y}}(x|y) \cdot p_{\mathcal{Y}}(y). \tag{5.5}$$

Неопределенность величины X при условии $Y = y$ определяется аналогично функции энтропии равенством

$$H(X|Y = y) = -\sum\nolimits_{i=1}^{n} p_{\mathcal{X}|\mathcal{Y}}(x|y) \cdot \log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y). \tag{5.6}$$

Эту величину можно интерпретировать как среднее количество информации, содержащейся в сообщении о значении случайной величины X, если уже известно, что $Y = y$.

Условной энтропией $H(X|Y)$ величины X по величине Y называется усредненное значение неопределенности $H(X|Y = y)$ по всем y. Т.е.

$$\begin{aligned} H(X|Y) &= \textstyle\sum_{y\in\mathcal{Y}} p_{\mathcal{Y}}(y) \cdot H(X|Y=y) = \\ &\overset{(5.6)}{=} -\textstyle\sum_{y\in\mathcal{Y}} p_{\mathcal{Y}}(y) \cdot \sum_{x\in\mathcal{X}} p_{\mathcal{X}|\mathcal{Y}}(x|y) \cdot \log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y) = \\ &= -\textstyle\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{\mathcal{Y}}(y) \cdot p_{\mathcal{X}|\mathcal{Y}}(x|y) \cdot \log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y) = \\ &\overset{(5.5)}{=} -\textstyle\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y) \cdot \log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y). \end{aligned} \tag{5.7}$$

Пусть $H(X,Y)$ определяется аналогично функции энтропии H от одной переменной.

Теорема 5.1 (*цепное правило*

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y).$$

Доказательство. Используя соотношения (5.5) и (5.7), получим

$$\begin{aligned} H(X,Y) &= -\textstyle\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y) \cdot \log_2 p_{\mathcal{X},\mathcal{Y}}(x,y) = \\ &= -\textstyle\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y) \cdot \log_2 p_{\mathcal{Y}}(y) - \\ &\quad -\textstyle\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y) \cdot \log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y) = \\ &= -\textstyle\sum_{y\in\mathcal{Y}} p_{\mathcal{Y}}(y) \cdot \log_2 p_{\mathcal{Y}}(y) + H(X|Y) \\ &= H(Y) + H(X|Y). \end{aligned}$$

Второе равенство следует из перестановочности аргументов в $H(X,Y)$. ∎

Иначе говоря, приведенная выше теорема утверждает, что неопределенность совместного распределения величин X и Y равна неопределенности X плюс непределенность Y по X.

Следствие 5.2
Пусть X и Y — независимые случайные величины. Тогда

i) $H(X,Y) = H(X) + H(Y)$,
ii) $H(X|Y) = H(X)$,
iii) $H(Y|X) = H(Y)$.

Доказательство. Для того, чтобы доказать утверждение i), повторим доказательство теоремы 5.1, используя равенство $p_{\mathcal{X},\mathcal{Y}}(x,y) = p_{\mathcal{X}}(x) \cdot p_{\mathcal{Y}}(y)$.

$$\begin{aligned} H(X,Y) = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 p_{\mathcal{X},\mathcal{Y}}(x,y) = \\ = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 p_{\mathcal{Y}}(y) - \\ & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 p_{\mathcal{X}}(x) = \\ = & -\sum_{y\in\mathcal{Y}} p_{\mathcal{Y}}(y)\cdot\log_2 p_{\mathcal{Y}}(y) - \sum_{x\in\mathcal{X}} p_{\mathcal{X}}(x)\cdot\log_2 p_{\mathcal{X}}(x) = \\ = & H(Y)+H(X). \end{aligned}$$

Утверждения ii) и iii) непосредственно следуют из i) и цепного правила. ■

Рассмотрим разность между количеством информации (см. (5.1)), содержащейся в сообщении $X = x$, и количеством информации, содержащейся в том же сообщении при заранее известном условии $Y = y$. Эту разность естественно считать количеством информации о том, что $X = x$, содержащейся в сообщении $Y = y$. Обозначим это через $I_{\mathcal{X};\mathcal{Y}}(x,y)$. Тогда выполняется равенство

$$I_{\mathcal{X};\mathcal{Y}}(x,y) = -\log_2 p_{\mathcal{X}}(x) - (-\log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y))$$

$$= -\log_2 \frac{p_{\mathcal{X}}(x)}{p_{\mathcal{X}|\mathcal{Y}}(x|y)} \overset{(5.5)}{=} -\log_2 \frac{p_{\mathcal{X}}(x)\cdot p_{\mathcal{Y}}(y)}{p_{\mathcal{X},\mathcal{Y}}(x,y)} = I_{\mathcal{Y};\mathcal{X}}(y,x).$$

Заметим, что имеет место симметрия $I_{\mathcal{X};\mathcal{Y}}(x,y) = I_{\mathcal{Y};\mathcal{X}}(y,x)$.

Взаимная информациия $I(X;Y)$ случайных величин X и Y определяется как математическое ожидание величины $I_{\mathcal{X};\mathcal{Y}}(x,y)$, т.е.

$$\begin{aligned} I(X;Y) = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot I_{\mathcal{X},\mathcal{Y}}(x,y) = \\ = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 \tfrac{p_{\mathcal{X}}(x)\cdot p_{\mathcal{Y}}(y)}{p_{\mathcal{X},\mathcal{Y}}(x,y)} = \\ = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 \tfrac{p_{\mathcal{X}}(x)}{p_{\mathcal{X}|\mathcal{Y}}(x|y)} = \\ = & I_{\mathcal{Y},\mathcal{X}}(y,x). \end{aligned} \tag{5.8}$$

Теорема 5.3

$$\begin{aligned} & I(X;Y) = H(X)+H(Y)-H(X,Y) = \\ & = H(X)-H(X|Y) = H(Y)-H(Y|X). \end{aligned}$$

Доказательство. Из равенства (5.8) следует, что

$$\begin{aligned} I(X;Y) = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 \tfrac{p_{\mathcal{X}}(x)}{p_{\mathcal{X}|\mathcal{Y}}(x|y)} = \\ = & -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 p_{\mathcal{X}}(x) + \\ & +\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p_{\mathcal{X},\mathcal{Y}}(x,y)\cdot\log_2 p_{\mathcal{X}|\mathcal{Y}}(x|y) = \\ = & -\sum_{x\in\mathcal{X}} p_{\mathcal{X}}(x)\cdot\log_2 p_{\mathcal{X}}(x) - H(X|Y) = \\ = & H(X)-H(X|Y). \end{aligned}$$

Остальные равенства следуют из теоремы 5.1.

■

Величину $I(X;Y)$ можно интерпретировать как среднее количество информации, которое Y дает об X (или, наоборот, X об Y).

Пример 5.3

Двоичный симметричный канал может быть описан следующим образом. Источник посылает значение величины X: $X = 0$ с вероятностью $1/2$ или $X = 1$ тоже с вероятностью $1/2$. Приемник получает значение величины Y: $Y = X$ с вероятностью $1 - p$ или же $Y = 1 - X$ с вероятностью p. Это означает, что $\mathcal{X} = \mathcal{Y} = \{0, 1\}$, и что

$$p_{\mathcal{Y}}(0) = p_{\mathcal{Y}|\mathcal{X}}(0|0)p_{\mathcal{X}}(0) + p_{\mathcal{Y}|\mathcal{X}}(0|1)p_{\mathcal{X}}(1) = (1-p)\cdot\frac{1}{2} + p\cdot\frac{1}{2} = \frac{1}{2}.$$

Аналогично, $p_{\mathcal{Y}}(1) = \frac{1}{2}$. Кроме того, $p_{\mathcal{X},\mathcal{Y}}(0,0) = p_{\mathcal{X},\mathcal{Y}}(1,1) = (1-p)/2$ и $p_{\mathcal{X},\mathcal{Y}}(0,1) = p_{\mathcal{X},\mathcal{Y}}(1,0) = p/2$. Таким образом, для двоичного симметричного канала по формуле (5.8) получаем

$$\begin{aligned} I(X;Y) &= -2\left\{\tfrac{1-p}{2}\log_2\tfrac{1/2}{1-p} + \tfrac{p}{2}\log_2\tfrac{1/2}{2}\right\} = \\ &= 1 + p\cdot\log_2 p + (1-p)\cdot\log_2(1-p) = 1 - H(p). \end{aligned}$$

Можно заключить, что приемник получает $1 - H(p)$ битов информации о случайной величине X по полученному значению величины Y. Практическое достижение "границы" $1-H(p)$ является фундаментальной проблемой алгебраической теории кодирования. См. [MacWS77], раздел 1.6.

При $p = 1/2$, как и следовало ожидать, приемник не получает никакой информации о переданных символах, поскольку $H(1/2) = 1$.

Вернемся к традиционной криптосистеме, описанной в гл. 1. Пусть выбор ключа из ключевого пространства $\mathcal{K}$ происходит с вероятностью $\Pr_{\mathcal{K}}(K = k)$, открытый текст представляет собой последовательность случайных величин

$$M^{(u)} = (M_0,\ M_1,\ \ldots,\ M_{u-1}),$$

а шифртекст — последовательность случайных величин

$$C^{(v)} = (C_0,\ C_1,\ \ldots,\ C_{v-1}).$$

Таким образом, $C^{(v)} = E_k(M^{(u)})$. В большинстве приложений v будет равно u. Поскольку E_k является взаимно-однозначным отображением, открытый текст должен однозначно определяться по ключу и шифртексту, следовательно, выполняется равенство

$$H(M^{(u)}|K, C^{(v)}) = 0. \tag{5.9}$$

Пользователя криптосистемы, конечно, интересует, как много информации об $M^{(u)}$ дает $C^{(v)}$.

Теорема 5.4

$$I(M^{(u)}; C^{(v)}) \geq H(M^{(u)}) - H(K)$$

Сформулируем это на словах. Неопределенность ключа в сумме с информацией об открытом тексте, которую дает шифртекст, не меньше, чем неопределенность открытого текста. Это вновь хорошо согласуется с нашей интуицией.

Доказательство теоремы 5.4. Из равенства (5.9) и цепного правила (теорему 5.1 можно применять и к условной энтропии) следует, что

$$H(K|C^{(v)}) = H(K|C^{(v)}) + H(M^{(u)}|K, C^{(v)}) = H(M^{(u)}, K|C^{(v)}) =$$

$$= H(M^{(u)}|C^{(v)}) + H(K|M^{(u)}, C^{(v)}) \geq H(M^{(u)}|C^{(v)}).$$

Т.е. неопределенность ключа по шифртексту не меньше, чем неопределенность открытого текста по шифртексту. И действительно, зная шифртекст, можно восстановить открытый текст при наличии ключа, но не всегда можно восстановить ключ при наличии открытого текста.

Из этого следует, что

$$H(M^{(u)}|C^{(v)}) \leq H(K|C^{(v)}) \leq H(K),$$

и по теореме 5.3 получаем

$$I(M^{(u)}; C^{(v)}) = H(M^{(u)}) - H(M^{(u)}|C^{(v)}) \geq H(M^{(u)}) - H(K).$$

■

Определение 5.3
Криптосистема называется *безусловно безопасной* или, как еще говорят, обладает *совершенной секретностью*, если

$$I(M^{(u)}; C^{(v)}) = 0.$$

Следствие 5.5
Если система абсолютно безопасна, то

$$H(M^{(u)}) \leq H(K).$$

Следствие 5.5 говорит, что в безусловно безопасной криптосистеме, в которой все ключи и все открытые тексты равновероятны, ключей должно быть не меньше, чем открытых текстов.

Пример 5.4

Допустим, что у нас 2^k ключей, и каждый из них может быть выбран с вероятностью $1/2^k$. Тогда

$$H(K) = -\sum_{i=1}^{2^k} \frac{1}{2^k} \log_2 \frac{1}{2^k} = k \text{ битов.}$$

Если сообщение представляет собой результат v бросаний правильной монеты, то, очевидно, $H(M^{(v)}) = v$, и для совершенной секретности необходимо выбрать $k \geq v$.

Это можно реализовать при шифровании $c^{(v)} = m^{(v)} \oplus k^{(v)}$, где через $k^{(v)}$ обозначены первые v битов ключа, а через $\oplus$ — покоординатное сложение по модулю 2. При таком шифровании каждый шифртекст $c^{(v)}$ может быть получен из каждого открытого текста с равной вероятностью.

5.3 Задачи

Задача 5.1. Покажите, что функция $-\sum_{i=1}^{n} p_i \cdot \log_2 p_i$ удовлетворяет условиям P1–P4 из разд. 5.1.

Задача 5.2. Пусть $\alpha \leq 1/2$.
a) Докажите, что

$$\frac{1}{n+1} \cdot \frac{n^n}{k^k (n-k)^{n-k}} \leq \binom{n}{k} \leq \frac{n^n}{k^k (n-k)^{n-k}}.$$

b) Используя эти неравенства, докажите, что

$$\lim_{x \to \infty} \frac{1}{n} \log \sum_{i=0}^{\lfloor \alpha n \rfloor} \binom{n}{i} = h(\alpha),$$

где через $h(x)$ обозначена функция энтропии, определенная формулой (5.4).

Задача 5.3. Предположим, что на одну букву аглийского текста приходится по полтора бита информации. Каково расстояние единственности для шифра Цезаря, примененного к английскому тексту? Ответьте на тот же вопрос для криптосистемы Виженера с ключом длины r.

Задача 5.4. Рассмотрим источник открытого текста без памяти, выдающий случайную величину X, равномерно распределенную на множестве $\{0, 1, 2\}$. На выходе канала приемник получает переменную Y, которая равна X с вероятностью $1 - p$, $0 \leq p \leq 1$, и принимает любое из двух оставшихся значений с вероятностью $p/2$. Вычислите взаимную информацию $I(X;Y)$ величин X и Y.

Задача 5.5. Пусть источник открытых текстов $\mathcal{S}$ выдает независимые друг от друга буквы X, одинаково распределенные на множестве $\{a, b, c, d\}$. Распределение вероятностей задается равенствами $\Pr(X = a) = 1/2$, $\Pr(X = b) = 1/4$ и $\Pr(X = c) = \Pr(X = d) = 1/8$. Рассмотрим две схемы кодирования:

схема A			схема B		
a	$\longmapsto$	00	a	$\longmapsto$	0
b	$\longmapsto$	01	b	$\longmapsto$	10
c	$\longmapsto$	10	c	$\longmapsto$	110
d	$\longmapsto$	11	d	$\longmapsto$	111

Получившийся открытый текст сначала преобразуется в последовательность из 0 и 1 по одной из приведенных выше схем, а затем шифруется с помощью алгоритма DES. Каково расстояние единственности для каждой из этих схем?

Задача 5.6. Докажите, что одноразовый щит является безусловно безопасной системой.

Глава 6

Техника сжатия данных

Из главы 5 (см. определения 5.1 и 5.2) становится понятно, что можно существенно повысить безопасность криптосистемы, если уменьшить избыточность открытого текста. В примере 5.1 показано как можно сократить избыточность.

В этой главе будут описаны два общих метода сокращения избыточности. Процесс удаления избыточности из текста называется *сжатием данных* или *кодированием источника.*

6.1 Основы кодирования стационарных источников

Пусть источник открытых текстов $\mathcal{S}$ выдает символы из алфавита $\{m_1, m_2, ..., m_n\}$ с вероятностями $p_1, p_2, ..., p_n$ независимо друг от друга. Символ m_i кодируется двоичной строкой $\underline{c}_i$ длины ℓ_i, $1 \leq i \leq n$.

Множество $\{\underline{c}_1, \underline{c}_2, ..., \underline{c}_n\}$ называется *кодом* C для источника $\mathcal{S}$. Идея процесса сжатия данных состоит в использовании кода, минимизирующего среднюю длину закодированного открытого текста. Поскольку источник порождает символы независимо друг от друга, достаточно будет минимизировать среднюю длину одного закодированного символа

$$L = \sum_{i=1}^{n} p_i \ell_i. \tag{6.1}$$

Минимизация производится за счет выбора наилучшего кода среди кодов C источника $\mathcal{S}$, обладающих одним дополнительным свойством. Приемник (декодер) должен быть в состоянии по полученной конкатенации кодовых слов однозначно восстановить посланное сообщение. Таким свойством обладают не все коды. Например, код $C = \{0, 01, 10\}$ не такой. Последовательность 010 в нем может получится двумя способами: 0, а следом 10, или 01, а следом 0. Таких двусмысленностей нужно избегать.

Определение 6.1
Код C называется *однозначно декодируемым* (сокращенно ОД-кодом), если каждая цепочка, полученная при конкатенации кодовых слов из C, однозначно разбивается на кодовые слова.

Пример 6.1

Пусть $n = 4$ и $C = \{0, 01, 011, 111\}$ (это код из примера 5.1 в обратном порядке). Покажем, что это ОД-код.

Рассмотрим конкатенацию кодовых слов. Если первый бит этой последовательности — 1, то первым кодовым словом должно быть 111. *Если же первый бит этой последовательности — 0, то либо последовательность имеет вид* $0\overbrace{11\ldots1}^{k}$ *для некоторого* $k \geq 0$, *либо в начале последовательности стоит подпоследовательность* $0\overbrace{11\ldots1}^{k}0$ *для некоторого* $k \geq 1$. *В зависимости от того, чему равно* k: 3ℓ, $3\ell+1$ *или* $3\ell+2$, *первым кодовым словом будет* 0, 01 *или* 011 *соответственно. После того, как первое кодовое слово однозначно определено, отделим его и применим описанное правило к оставшейся более короткой двоичной последовательности.*

Теорема 6.1 (*неравенство Мак-Миллана*) [McMi56]
Однозначно декодируемый код C мощности n с длинами кодовых слов ℓ_i, $1 \leq i \leq n$, существует тогда и только тогда, когда

$$\sum_{i=1}^{n} \frac{1}{2^{\ell_i}} \leq 1. \tag{6.2}$$

Доказательство. Докажем лишь то, что данное неравенство является необходимым условием существования ОД-кода с кодовыми словами c_i длины ℓ_i, $1 \leq i \leq n$. То, что оно является также и достаточным условием, будет доказано позже в этой же главе. Пусть $L = \sum_{i=1}^{n} \frac{1}{2^{\ell_i}}$. Без ограничения общности можно считать, что $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_n$. Тогда

$$L^N = \left(\sum\nolimits_{i=1}^{n} \frac{1}{2^{\ell_i}}\right)^N = \sum\nolimits_{j=N\cdot\ell_1}^{j=N\cdot\ell_n} \frac{A_j}{2^j}, \tag{$\star$}$$

где через A_j обозначено число способов представления числа j в виде суммы $\ell_{i_1} + \ell_{i_2} + \ldots + \ell_{i_N}$. Иначе говоря, A_j — это число способов составления конкатенации длины j из N кодовых слов.

Двоичных последовательностей длины j всего 2^j штук. Поскольку C — ОД-код, никакая из этих последовательностей не может быть представлена в виде некоторой конкатенации кодовых слов более, чем одним способом, и, тем более, в виде конкатенации ровно N кодовых слов. Следовательно, $A_j \leq 2^j$. Подставив эти неравенства в формулу ($\star$), получим, что для всех $N \geq 1$ выполняется неравенство

$$L^N \leq \sum\nolimits_{j=N\cdot\ell_1}^{j=N\cdot\ell_n} 1 = N(\ell_n - \ell_1) + 1.$$

Предположив, что $L > 1$, получим, что экспоненциально растущая функция от N оценивается сверху линейной функцией от N. Это противоречие показывает, что $L \leq 1$.

■

В примере 6.1 для декодирования полученной последовательности пришлось исследовать ее префикс, превосходящий по длине самое длинное кодовое слово. С практической точки зрения это неудобно.

Определение 6.2
Код C называется *префиксным* или *мгновенным* кодом, если никакое его кодовое слово не является префиксом другого.

Код из примера 6.1 не является префиксным, потому что, например, кодовое слово 0 — префикс кодового слова 01. Ясно, что код из примера 5.1 префиксный. При декодирования последовательности, полученной сцеплением кодовых слов префиксного кода, просто отыщем какой-то ее префикс в списке кодовых слов. Из-за того, что код префиксный, это можно сделать единственным способом. Удалив найденное слово, продолжим точно так же.

Отметим, что при использовании префиксного кода для определения первого кодового слова требуется просматривать не более чем ℓ_n первых символов декодируемой последовательности.

Проведенное наблюдение доказывает следующую лемму.

Лемма 6.2
Любой префиксный код является однозначно декодируемым.

Теорема 6.3 (*неравенство Крафта* [Kraf49])
Префиксный код C мощности n с длинами кодовых слов ℓ_i, $1 \le i \le n$, существует тогда и только тогда, когда

$$\sum_{i=1}^{n} \frac{1}{2^{\ell_i}} \le 1. \tag{6.3}$$

Доказательство. По лемме 6.2 префиксный код однозначно декодируем. Следовательно, из неравенства Мак-Миллана (теорема 6.1) следует, что неравенство (6.3) выполняется для каждого префиксного кода.

Покажем теперь, что для каждого набора из n чисел ℓ_i, $1 \le i \le n$, существует префиксный код (а, значит, и ОД-код) с кодовыми словами $\underline{c}_i$ длины ℓ_i, $1 \le i \le n$. Без ограничения общности можно считать, что $\ell_1 \le \ell_2 \le \ldots \le \ell_n$. Благодаря такому упорядочению и неравенству $\sum_{i=1}^{n-1} \frac{1}{2^{\ell_i}} < 1$, можно взять в качестве векторов $\underline{c}_i = (c_{i,1}, c_{i,2}, \ldots, c_{i,\ell_i})$, $1 \le i \le n$, двоичные представления чисел $\sum_{j=1}^{i-1} \frac{1}{2^{\ell_j}}$. Т.е.

$$\sum_{j=1}^{i-1} \frac{1}{2^{\ell_j}} = \frac{c_{i,1}}{2} + \frac{c_{i,2}}{2^2} + \ldots + \frac{c_{i,\ell_i}}{2^{\ell_i}}.$$

Например, $\underline{c}_1 = (0, 0, \ldots, 0)$ длины ℓ_1, $\underline{c}_2 = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$ длины ℓ_2 с 1 в позиции с номером ℓ_1, $\underline{c}_3 = (0, 0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0)$ длины ℓ_3 с 1 в позициях с номерами ℓ_1 и ℓ_2, если $\ell_1 \ne \ell_2$, а если $\ell_1 = \ell_2$, то

$\underline{c}_3 = (0,0,...,0,1,0,...,0)$ длины ℓ_3 с 1 в позиции с номером $\ell_1 - 1$, и т.д. По определению длина кодового слова $\underline{c}_i$ равна ℓ_i.

Осталось показать, что никакое кодовое слово $\underline{c}_u$ не может быть префиксом кодового слова $\underline{c}_v$ при $u \neq v$. Предположим противное. Ясно, что $\ell_u \neq \ell_v$, иначе эти два слова просто совпадают. Пусть $\ell_u < \ell_v$, тогда $u < v$, из чего следует, что

$$\sum_{j=1}^{v-1} \frac{1}{2^{\ell_j}} - \sum_{j=1}^{u-1} \frac{1}{2^{\ell_i}} \overset{\text{опр.}}{=} \sum_{j=1}^{\ell_v} \frac{c_{v,j}}{2^{\ell_j}} - \sum_{j=1}^{\ell_u} \frac{c_{u,j}}{2^{\ell_j}} \overset{\text{префикс}}{=}$$

$$= \sum_{j=\ell_u+1}^{\ell_v} \frac{c_{v,j}}{2^{\ell_j}} \leq \sum_{j=\ell_u+1}^{\ell_v} \frac{1}{2^{\ell_j}} < \sum_{j=\ell_u+1}^{\infty} \frac{1}{2^{\ell_j}} = \frac{1}{2^{\ell_u}}.$$

С другой стороны,

$$\sum_{j=1}^{v-1} \frac{1}{2^{\ell_j}} - \sum_{j=1}^{u-1} \frac{1}{2^{\ell_i}} = \sum_{j=u}^{v-1} \frac{1}{2^{\ell_j}} \geq \frac{1}{2^{\ell_u}}.$$

Два этих неравенства противоречат друг другу.

■

Пример 6.2

Рассмотрим числа $\ell_1 = 1$, $\ell_2 = 2$, $\ell_3 = 3$ *и* $\ell_4 = \ell_5 = 4$.
Поскольку $\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^4} = 1$, *неравенство Крафта выполняется. Приведенное выше доказательство дает следующие кодовые слова (используются функции* Length, Do, Table, IntegerDigits *и* Print *пакета "Mathematica"):*

```
l = {1, 2, 3, 4, 4};
L = Length[l]; c=.;
c[1] = Table[0, {l[[1]]}];
Do[c[i] = IntegerDigits[Ceiling[(Sum[1/2^l[[j]], {j, 1, i-1}]) 2^l[[j]], 2], {i, 2, L}];
Do[Print[c[i]], {i, 2, L}]
```

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

Легко проверить, что этот код префиксный.

Как замечательно, что неравенства Мак-Миллана и Крафта ((6.2) и (6.3)) совпадают. Это означает, что минимум математических ожиданий длин ОД-кодов равен минимуму математических ожиданий длин префиксных кодов!

Две следующие теоремы дают оценки для средней длины префиксного кода (или ОД-кода).

Теорема 6.4
Рассмотрим источник открытых текстов $\mathcal{S}$, выдающий символы m_i с вероятностями p_i, $1 \le i \le n$. Пусть ОД-код C отображает символ m_i в двоичную строку $\underline{c}_i$ длины ℓ_i, $1 \le i \le n$. Тогда средняя длина кодового слова $L = \sum_{i=1}^{n} p_i \cdot \ell_i$ удовлетворяет условию

$$L \ge H(\underline{p}).$$

Доказательство. Из известного неравенства $\ln x \le x - 1$, $x > 0$ и неравенства (6.2) следует, что

$$H(\underline{p}) - L = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i - \sum_{i=1}^{n} p_i \ell_i = \frac{1}{\ln 2} \sum_{i=1}^{n} p_i \cdot \ln \frac{1}{p_i \cdot 2^{\ell_i}} \le$$

$$\le \frac{1}{\ln 2} \sum_{i=1}^{n} p_i \left(\frac{1}{p_i \cdot 2^{\ell_i}} - 1 \right) = \frac{1}{\ln 2} \left(\left(\sum_{i=1}^{n} \frac{1}{2^{\ell_i}} \right) - 1 \right) \le 0.$$

■

Теорема 6.5
Пусть источник открытых текстов $\mathcal{S}$ выдает символы m_i с вероятностями p_i, $1 \le i \le n$. Тогда существует соответствующий этому источнику префиксный код C, у которого средняя длина кодового слова L удовлетворяет условию

$$L < H(\underline{p}) + 1.$$

Доказательство. Пусть $\ell_i = \lceil \log_2 1/p_i \rceil$, $1 \le i \le n$. Тогда $2^{\ell_i} \ge 1/p_i$ и, значит,

$$\sum_{i=1}^{n} 1/2^{\ell_i} \le \sum_{i=1}^{n} p_i = 1.$$

Для этих величин ℓ_i, $1 \le i \le n$, построим код C так, как описано в доказательстве теоремы 6.3. Это префиксный код, и средняя длина его кодового слова L удовлетворяет условию

$$L = \sum_{i=1}^{n} p_i \cdot \ell_i = \sum_{i=1}^{n} p_i \cdot \lceil \log_2 1/p_i \rceil < \sum_{i=1}^{n} p_i \cdot (\log_2 1/p_i + 1) = H(\underline{p}) + 1.$$

■

Следствие 6.6
Минимум средней длины L префиксного (или ОД) кода, соответствующего источнику открытых текстов $\mathcal{S}$ с распределением вероятностей $\underline{p}$, удовлетворяет условию

$$H(\underline{p}) \le L < H(\underline{p}) + 1.$$

Энтропия последовательности из N независимо порождаемых символов ровно в N раз превосходит энтропию одного символа. Применим

следствие 6.6 к N-кам символов, выдаваемых источником. Получим, что минимальное математическое ожидание длины закодированной N-граммы $L^{(N)}$ удовлетворяет условию

$$N \cdot H(\underline{p}) \leq L^{(N)} < N \cdot H(\underline{p}) + 1.$$

Это означает, что

$$H(\underline{p}) \leq \frac{L^{(N)}}{N} < H(\underline{p}) + \frac{1}{N}. \tag{6.4}$$

Таким образом, $\lim_{N\to\infty} \frac{L^{(N)}}{N} = H(p)$. Этим подтверждается последняя из трех приведенных в начале гл. 5 интерпретаций функции энтропии H.

Теперь выведем несколько свойств, которыми обладает префиксный код с минимальной величиной L.

Теорема 6.7
Рассмотрим источник открытых текстов $\mathcal{S}$, выдающий символы m_i, $1 \leq i \leq n$, с вероятностями $p_1 \geq p_2 \geq \ldots \geq p_n$. Пусть C — ОД-код с минимально возможным для такого источника математическим ожиданием длины кодового слова L. По-прежнему его кодовые слова обозначаются через $\underline{c}_i$, а их длины через ℓ_i, $1 \leq i \leq n$. Тогда после подходящего перенумерования кодовых слов, соответствующих равновероятным символам, будут выполняться следующие условия:

P1) $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_n$.
P2) Можно считать, что код C префиксный.
P3) $\sum_{i=1}^{n} \frac{1}{2^{\ell_i}} = 1$.
P4) $\ell_{n-1} = \ell_n$.
P5) Есть два кодовых слова длины ℓ_n, отличающихся друг от друга только последними символами.

Доказательство

P1) Допустим, что $p_u > p_v$ и $\ell_u > \ell_v$. Изготовим из C новый код C^*, поменяв местами слова $\underline{c}_u$ и $\underline{c}_v$. Тогда C^* тоже будет однозначно декодируемым. А средняя длина L^* слова кода C^* будет равна

$$L^* = L + p_u(\ell_v - \ell_u) + p_v(\ell_u - \ell_v) = L + (p_u - p_v)(\ell_v - \ell_u) < L.$$

Это противоречит предположению о минимальности L.

Если $p_u = p_v$, $u < v$, неравенство $\ell_u \leq \ell_v$ можно получить после необходимого перенумерования равновероятных кодовых слов.

P2) Поскольку необходимые и достаточные условия теорем 6.1 и 6.2 одинаковы, для каждого ОД-кода существует префиксный код с теми же длинами кодовых слов, а, значит, и той же средней длиной кодового слова L.

P3) Если $\sum_{i=1}^{n} \frac{1}{2^{\ell_i}} < 1$, то $\sum_{i=1}^{n} \frac{1}{2^{\ell_i}} \leq \frac{2^{\ell_n}-1}{2^{\ell_n}}$, поэтому после уменьшения ℓ_n на 1 неравенство Крафта (6.3) для кода по-прежнему будет выполняться. По теореме 6.2 найдется префиксный код со средней длиной кодового слова меньше L, что противоречит предположению о минимальности L.

P4) Если $\ell_n > \ell_{n-1}$, то из P1 следует, что слово $\underline{c}_n$ строго длиннее всех остальных. Тогда левая часть равенства в P3 представляет собой рациональное число со знаменателем 2^{ℓ_i} и нечетным числителем. Такое число не может равняться 1.

P5) Удалим последний символ в кодовом слове $\underline{c}_n$ и обозначим получившийся двоичный вектор через $\underline{c}_n^*$. Пусть $C^* = \{\underline{c}_1, \underline{c}_2, ..., \underline{c}_{n-1}, \underline{c}_n^*\}$. Из P3 следует, что код C^* не удовлетворяет неравенству Крафта (6.3), значит, не может быть префиксным, каковым по P2 можно считать код C. Это может означать только то, что слово $\underline{c}_n^*$ является собственным префиксом некоторого слова $\underline{c}_j$, для $1 \leq j \leq n-1$. Следовательно, длина слова $\underline{c}_j$ равна ℓ_n, а также слова $\underline{c}_n$ и $\underline{c}_j$ отличаются только последними символами.

■

Свойство P5 дает ключ к методу построения ОД-кода с минимальным математическим ожиданием длины кодового слова. Этот метод будет описан в следующем разделе.

6.2 Коды Хаффмана

Алгоритм Хаффмана [Huff52] строит для каждого источника открытых текстов со стационарным распределением префиксный код с минимальным для данного источника математическим ожиданием длины кодового слова. Этот алгоритм рекурсивный.

Если источник открытых текстов может выдавать с ненулевой вероятностью только два символа, то наилучшим кодированием будет замена одного из них на 0, а второго на 1. Ясно, что в таком случае $L < H(\underline{p}) + 1$.

Каждый шаг рекурсии состоит из двух частей: процесса слияния (редуцирования) и процесса расщепления.

Процесс слияния.

Пусть источник открытых текстов $\mathcal{S}$ выдает символы m_i, $1 \leq i \leq n$, с вероятностями $p_1 \geq p_2 \geq ... \geq p_n$. Заменим символы m_{n-1} и m_n одним новым символом m_{n-1}^*, вероятность появления которого равна $p_{n-1}^* = p_{n-1} + p_n$. В алфавите получившегося нового источника $\mathcal{S}^*$ на один символ меньше, чем в алфавите $\mathcal{S}$.

Процесс расщепления.

Пусть источник $\mathcal{S}^*$ генерирует символы алфавита $\{\underline{m}_1, \underline{m}_2, ..., \underline{m}_{n-2}, \underline{m}_{n-1}^*\}$. Пусть $C^* = \{\underline{c}_1, \underline{c}_2, ..., \underline{c}_{n-2}, \underline{c}_{n-1}^*\}$ — префиксный код с минимальным для источника $\mathcal{S}^*$ математическим ожиданием длины кодового сло-

ва (можно считать, что при построении этого кода в процессе рекурсии символы были перенумерованы в порядке невозрастания вероятностей их появления).

Код C определяется равенствами

$$\underline{c}_i = \underline{c}_i^* \text{ для } 1 \le i \le n-2,$$

$$\underline{c}_{n-1} = (\underline{c}_{n-1}^*, 0),$$

$$\underline{c}_n = (\underline{c}_{n-1}^*, 1),$$

т.е. при расщеплении символа $\underline{m}_{n-1}^*$ на два символа $\underline{m}_{n-1}$ и $\underline{m}_n$ кодовые слова для них получаются в результате приписывания, соответственно, 0 и 1 к слову $\underline{c}_{n-1}^*$.

Пример 6.3

Пусть $n = 6$ и источник S генерирует символы с вероятностями, указанными в следующей таблице:

m_1	m_2	m_3	m_4	m_5	m_6
0.3	0.2	0.2	0.1	0.1	0.1

Для того, чтобы запомнить путь, по которому идет процесс слияния, будем обозначать символ m_{n-1}^* через $(m_{n-1} + m_n)$. После одного слияния и упорядочения символов в порядке невозрастания вероятностей их появления получим

m_1	m_2	m_3	$(m_5 + m_6)$	m_4
0.3	0.2	0.2	0.2	0.1

Повторяя эту процедуру, получаем

m_1	$(m_4 + (m_5 + m_6))$	m_2	m_3
0.3	0.3	0.2	0.2

затем

$(m_2 + m_3)$	m_1	$(m_4 + (m_5 + m_6))$
0.4	0.3	0.3

и, наконец,

$(m_1 + (m_4 + (m_5 + m_6)))$	$(m_2 + m_3)$
0.6	0.4

В процессе расщепления проделаем описанные выше шаги в обратном порядке. Начинаем с кода $\{0, 1\}$. На каждом шаге один символ заменяется двумя новыми, коды которых получаются приписыванием к коду старого символа 0 и 1 соответственно.

Отметим, что на каждом шаге m_i заменяется на $\underline{c}_i$. Получаем цепочку

$(\underline{c}_1 + (\underline{c}_4 + (\underline{c}_5 + \underline{c}_6)))$	$(\underline{c}_2 + \underline{c}_3)$
(0)	(1)

затем

$(\underline{c}_2 + \underline{c}_3)$	$\underline{c}_1$	$(\underline{c}_4 + (\underline{c}_5 + \underline{c}_6))$
(1)	(0,0)	(0,1)

затем

$\underline{c}_1$	$(\underline{c}_4 + (\underline{c}_5 + \underline{c}_6))$	$\underline{c}_2$	$\underline{c}_3$
(0,0)	(0,1)	(1,0)	(1,1)

затем

$\underline{c}_1$	$\underline{c}_2$	$\underline{c}_3$	$(\underline{c}_5 + \underline{c}_6)$	$\underline{c}_4$
(0,0)	(1,0)	(1,1)	(0,1,0)	(0,1,1)

и, наконец, код источника $\mathcal{S}$:

$\underline{c}_1$	$\underline{c}_2$	$\underline{c}_3$	$\underline{c}_4$	$\underline{c}_5$	$\underline{c}_6$
(0,0)	(1,0)	(1,1)	(0,1,1)	(0,1,0,0)	(0,1,0,1)

Видно, что $\ell_1 = \ell_2 = \ell_3 = 2$, $\ell_4 = 3$ и $\ell_5 = \ell_6 = 4$. Легко проверить, что $\sum_{i=1}^{6} 1/2^{\ell_i} = 1$ и, что $H(\underline{p}) \leq L < H(\underline{p}) + 1$. Применим функцию MultiEntropy, определенную в разд. 5.1, а затем функцию Length пакета "Mathematica".

```
MultiEntropy[p_List] :=- Sum[p[[i]]*Log[2,p[[i]]], {i, 1, Length[p]}]
```

```
p = {0.3, 0.2, 0.2, 0.1, 0.1, 0.1};
MultiEntropy[p]
l = {2, 2, 2, 3, 4, 4}; len = Length[l];
Sum[1/2^l[[j]], {i, 1, len}] == 1
Sum[p[[i]]*l[[i]], {i, 1, len}]
```

```
2.44644
```

```
True
```

```
2.5
```

Для демонстрации действия этого кода Хаффмана применим его к тексту, составленному из 6 первых букв алфавита. Сначала смоделируем источник с помощью функций Which, Random и Do пакета "Mathematica" (знаком <> обозначается конкатенация цепочек).

```
SeedRandom[12321]; randomchar[x_] :=
Which[x < 0.3, "a", x < 0.5, "b", x < 0.7, "c",
x < 0.8, "d", x < 0.9, "e", x < 1, "f"];
sourcetext = ""; n = 10;
Do[sourcetext =
sourcetext <> randomchar[Random[Real, {0, 1}]], {j, 1, n}];
sourcetext
```

```
|| eedcbccaec
```

Для кодирования применим описанный выше код Хаффмана и функцию StringReplace пакета "Mathematica".

```
code = StringReplace[sourcetext, {"a" → "00", "b" → "10",
"c" → "11", "d" → "011", "e" → "0100", "f" → "0101"}]
```

```
|| 01000100011111010111100010011
```

Чтобы сравнить длину такого кодирования с энтропией источника, применим определенную выше функцию MultiEntropy и функцию StringLength пакета "Mathematica".

```
StringLength[code] / n - MultiEntropy[p]
```

```
|| 0.253561
```

Процесс декодирования в пакете "Mathematica" можно выполнить с помощью функции StringReplace, поскольку эта функция действует слева направо.

```
StringReplace[code, {"0101" → "f",
"0100" → "e", "011" → "d", "11" → "c", "10" → "b",
"00" → "a"}]
sourcetext == st
```

```
|| eedcbccaec

|| True
```

Лучший способ описания процесса декодирования дает следующий рисунок. Читаем полученную строку слева направо. Одновременно с этим перемещаемся по дереву, начиная от корня. В зависимости от того, прочитан символ 0 или символ 1, сдвигаемся по дереву вправо вниз или вправо вверх. При достижении листа дерева (концевой вершины) пишем соответствующий этому листу символ алфавита источника, после чего вновь начинаем от корня. Например, два первых символа в последовательности "0001000001000010100001001 1" это "00". Их прочтение приводит к символу "a". Прочтение следующих четырех символов "0100" приводит к символу "e", и т.д.

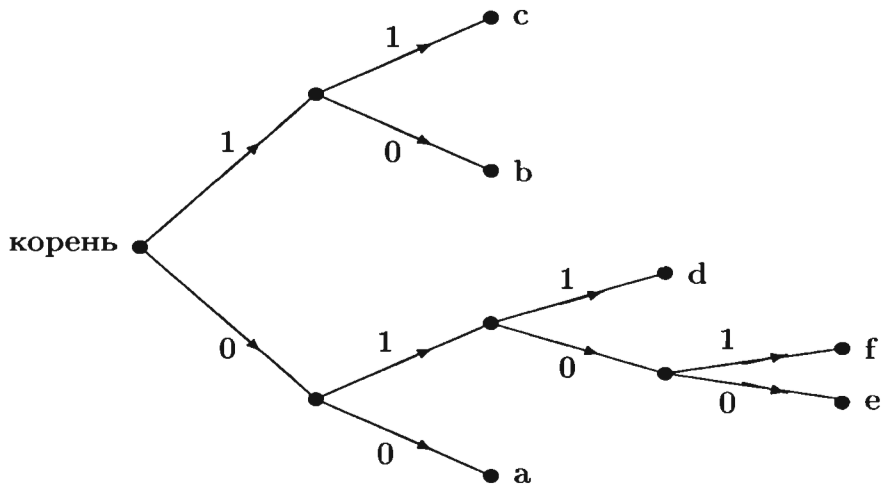


Рис. 6.1. Дерево декодирования для кода Хаффмана.

Лемма 6.8
Пусть источник открытых текстов $\mathcal{S}$ выдает символы m_i, $1 \le i \le n$, с вероятностями $p_1 \ge p_2 \ge \ldots \ge p_n$. Обозначим через $\mathcal{S}^*$ редуцированный источник открытых текстов, который выдает символы m_i^*, $1 \le i \le n-1$, с вероятностями $p_i^* = p_i$, $1 \le i \le n-2$, и $p_{n-1}^* = p_{n-1} + p_n$. Предположим, что C^* — префиксный код с минимальным для источника $\mathcal{S}^*$ математическим ожиданием длины кодового слова. Пусть слова из C^* обозначаются через $\underline{c}_i^*$, $1 \le i \le n-1$. Определим для источника $\mathcal{S}$ код C так, что $\underline{c}_i = \underline{c}_i^*$ для $1 \le i \le n-2$, $\underline{c}_{n-1} = ((\underline{c}_{n-1}^*)_1, \ldots, (\underline{c}_{n-1}^*)_{n-1}, 0)$, и $\underline{c}_n = ((\underline{c}_{n-1}^*)_1, \ldots, (\underline{c}_{n-1}^*)_{n-1}, 1)$. Тогда C — префиксный код с минимальным для источника $\mathcal{S}$ математическим ожиданием длины кодового слова.

Доказательство. Очевидно, что код C префиксный. Через l_i и l_i^* обозначим длины слов $\underline{c}_i$ и $\underline{c}_i^*$ соответственно. Эти числа связаны равенствами $l_i = l_i^*$ при $1 \le i \le n-2$ и $l_{n-1} = l_n = l_{n-1}^* + 1$. Средние длины L и L^* кодов C и, соответственно, C^* связаны равенством

$$L = \sum_{i=1}^{n} p_i l_i = \sum_{i=1}^{n-2} p_i l_i + p_{n-1} l_{n-1} + p_n l_n = \sum_{i=1}^{n-2} p_i^* l_i^* + p_{n-1}(l_{n-1}^* + 1) +$$

$$+ p_n(l_{n-1}^* + 1) = \sum_{i=1}^{n-2} p_i^* l_i^* + (p_{n-1} + p_n) l_{n-1}^* + (p_{n-1} + p_n) =$$

$$= \sum_{i=1}^{n-2} p_i^* l_i^* + p_{n-1}^* l_{n-1}^* + (p_{n-1} + p_n) = L^* + (p_{n-1} + p_n).$$

Из теоремы 6.7 и рассуждения, подобного приведенному выше, следует, что любой префиксный код $\hat{C}$ с минимальным для источника $\mathcal{S}$ математическим ожиданием длины кодового слова сводится к префиксному коду для источника $\mathcal{S}^*$ со средней длиной кодового слова, равной

$\hat{L} - (p_{n-1} + p_n)$. Поскольку длина L^* была минимальна для $\mathcal{S}^*$, имеем $\hat{L} - (p_{n-1} + p_n) \geq L^* = L - (p_{n-1} + p_n)$, т.е. $\hat{L} \geq L$. Но длина $\hat{L}$ была минимальна для $\mathcal{S}$, следовательно, $\hat{L} = L$, т.е. код C имеет минимальное для источника $\mathcal{S}$ математическое ожидание длины кодового слова.

■

Лемма 6.9
Пусть источник открытых текстов $\mathcal{S}$ выдает символы m_i, $1 \leq \leq i \leq n$, с вероятностями $p_1 \geq p_2 \geq \ldots \geq p_n$. Тогда код Хаффмана имеет минимальное среди всех ОД-кодов для данного источника математическое ожидание L длины кодового слова.

Доказательство. При $n = 2$ утверждение очевидно, поскольку код Хаффмана будет равен $\{(0), (1)\}$ и $L = 1$. Индуктивный переход — это прямое следствие из леммы 6.8.

■

6.3 Универсальное сжатие данных, алгоритмы Лемпеля-Зива

Если требуется сжимать данные из источника с неизвестным распределением, то алгоритм Хаффмана оказывается неприменимым. В такой ситуации необходима так называемая техника *универсального сжатия данных*. Примерами такой техники являются алгоритмы Лемпеля-Зива (их два) и техника, называемая арифметическим кодированием (см. [ZivL77], [ZivL78] и, соответственно, [RisL79]).

В [ZivL77] авторы вводят окно фиксированной длины, которое скользит, скажем, слева направо над последовательностью символов, порожденной источником. Это *скользящее окно* состоит из двух частей: бóльшая часть слева, называемая *буфером поиска*, и меньшая часть справа, называемая *буфером предварительного просмотра*. Символы источника в буфере поиска уже закодированы. Кодер ищет в буфере поиска самую длинную из копий префиксов буфера предварительного просмотра и кодирует этот префикс. Предположим, что первые j еще не закодированных символов совпали с j символами в буфере поиска, расположенными, начиная с i-й его позиции, а вслед за ними стоит символ a, не совпадающий со следующим символом в буфере поиска. Тогда кодер выдаст тройку (i, j, a), а скользящее окно сдвинется на $j + 1$ букву вправо.

Например, предположим, что длина буфера поиска равна 10, а длина буфера предварительного просмотра — 5. Пусть скользящее окно выглядит так, как показано на рис. 6.2.

Самой длинной парой совпадающих кусков, которые можно найти, будут первые три символа буфера предварительного просмотра и три буквы, начиная с 3-й позиции буфера поиска. Кодер пошлет тройку $(3, 3, a)$, где a — первый символ, с которого начинается несовпадение.

Рис. 6.2. Скользящее окно.

Скользящее окно сдвинется на четыре символа вправо. В начале работы, когда буфер поиска пуст, кодер выдает $(0, 0, x)$, где x — первый символ, выданный источником.

Обсудим теперь частный случай кодов *Лемпеля-Зива*. Мы следуем [Well99], где также можно найти и анализ этого их варианта. Основная его идея состоит в том, что обе стороны (отправитель и получатель) создают словарь, в котором в компактном виде представлены подстроки, пересылавшиеся ранее. Если потребуется сжать новую строку, которая уже содержится в словаре, то ее кодируют при помощи индекса соответствующей записи в словаре. В общем случае этот индекс оказывается намного короче, чем сама строка. Если же новая строка еще не содержится в словаре, то потребуется выполнить более значительную работу.

Словарь, который одновременно создают отправитель и получатель, будет (значительно) больше, чем алфавит $\mathcal{A}$ источника $\mathcal{S}$. Однако его можно хранить весьма эффективным способом в виде так называемого связного списка.

Читатель должен понимать, что использование алгоритма Лемпеля-Зива включает кое-что большее. Однако, для файлов средней длины (скажем, размером в одну страницу текста) его применение уже оправдано.

□ Инициализация

Как уже отмечалось выше, *словарь* хранится в виде *связного списка*. Каждая запись в этом списке имеет собственный *адрес и*. Запись представляет собой пару (v, a), в которой v — указатель на другую запись словаря (т.е. v — какой-то адрес), а a — буква алфавита $\mathcal{A}$. Обозначим мощность алфавита $\mathcal{A}$ через A.

Перед запуском алгоритма заполним словарь следующими записями (их $A + 1$ штук):

адрес	указатель	буква
0	0	$\emptyset$
1	0	a_1
2	0	a_2
⋮	⋮	⋮
A	0	a_A

Заметьте, что все эти записи указывают на элемент списка с адресом 0. В алфавите $\mathcal{A}$ нет символа $\emptyset$. Это дополнительный символ, служащий знаком препинания.

Чтобы запустить алгоритм, сделаем значение указателя v равным 0, а значение указателя адреса u равным $A+1$ (u содержит адрес первой пустой записи в связном списке).

□ Кодирование

Алгоритм 6.10 (*Кодирование Лемпеля-Зива*)
do **begin** прочитать следующий символ источника a
if пара (v,a) содержится в словаре
then присвоить переменной v значение адреса пары (v,a)
else **begin**
1) переслать v,
2) поместить в словарь новую запись (v,a) по адресу u,
3) $u := u+1$ (увеличить указатель u на 1),
4) присвоить переменной v значение адреса пары $(0,a)$
end
until остановка работы источника[1]

Процедура, описанная выше, имеет следующую интерпретацию. Если пара (v,a) содержится в словаре, то кодер перерабатывает строку, которая уже встречалась когда-то ранее. Приписывание переменной v значения адреса пары (v,a) создает возможность позднее восстановить этот список.

Если пара (v,a) не содержится в словаре, то кодер столкнулся со строкой, которую он ранее не перерабатывал. Он пересылает v, по которому получатель узнает адрес последнего символа предшествующей строки. Далее кодер добавляет в словарь новую запись (v,a) по адресу u. Символ a будет служить корнем новой строки. Указателю v придается значение адреса записи $(0,a)$. Число 0 в этой записи указывает в словаре на запись $(0,\emptyset)$, которая отмечает начало новой строки.

Заметим, что выходными символами в процессе кодирования являются словарные индексы, точнее, адреса связного списка. Их длина растет как логарифм от длины словаря. Заметим также, что все более часто при прочтении нового символа источника не генерируется новый выходной символ, поскольку текущая строка уже кодировалась ранее.

Пример 6.4 (часть 1)

Допустим, что мы хотим сжать двоичную строку $\{s_i\}_{i=1}^n$, *т.е.* $A=\{0,1\}$ *и* $A=2$. *Инициализируем процесс кодирования, положив*

```
Dict = {{0, -1}, {0, 0}, {0, 1}}
u = 3; v = 0; output {};
```

```
|| {{0, -1}, {0, 0}, {0, 1}}
```

Отметим, что вместо пустого символа $\emptyset$ *используется отрицательное число* -1.

[1] После этого требуется еще раз переслать v. — *Прим. перев.*

Для демонстрации процесса кодирования будем для каждого нового символа s_i источника выдавать новый словарь (представленный в виде связного списка), новые значения переменных u и v и полную выходную последовательность.

Применим функцию Position пакета "Mathematica", которая находит место элемента в списке. Поскольку наш список включает в себя списки в качестве элементов, мы добавляем [[1]] дважды. Заметьте, что мы вычитаем 1 из адреса, поскольку наша нумерация начинается с 0, а не с 1.

```
Pos[s_List, el_List] := Position[s, el] [[1]] [[1]] - 1
```

Например,

```
l = {{3}, {5}, {7}, {2}, {1}};
el = {7};
Pos[l, el]
```

```
|| 2
```

Теперь мы готовы к процессу кодирования. Применим функции Do, If, MemberQ, Append и Print пакета "Mathematica".

```
s = {1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1};
Do[If[MemberQ[Dict, {v, s[[i]]}],
       v = Pos[Dict, {v, s[[i]]}],
output = Append[output,v];
Dict = Append[Dict, {v, s[[i]]}];
v = Pos[Dict, {0, s[[i]]}]];
Print[Dict, ", v=", v, ", total output is ", output],
{i, 1, Length[s]}]
```

[2]

```
|| {{0, -1}, {0, 0}, {0, 1}}, v=2, total output is {}

|| {{0, -1}, {0, 0}, {0, 1}, {2, 1}},
|| v=2, total output is {2}

|| {{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}},
|| v=1, total output is {2, 2}

|| {{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}},
|| v=1, total output is {2, 2, 1}

|| {{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}},
|| v=5, total output is {2, 2, 1}

|| {{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0},
|| {5, 1}}, v=2, total output is {2, 2, 1, 5}
```

[2] Следует добавить команду `output = Append[output,v]`. — *Прим. перев.*

```
{{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0},
{5, 1}}, v=4, total output is {2, 2, 1, 5}

{{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}, {5, 1},
{4, 1}}, v=2, total output is {2, 2, 1, 5, 4}

{{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}, {5, 1},
{4, 1}}, v=3, total output is {2, 2, 1, 5, 4}

{{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}, {5, 1},
{4, 1}, {3, 0}}, v=1, total output is {2, 2, 1, 5, 4, 3}

{{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}, {5, 1},
{4, 1}, {3, 0}}, v=5, total output is {2, 2, 1, 5, 4, 3}

{{0, -1}, {0, 0}, {0, 1}, {2, 1}, {2, 0}, {1, 0}, {5, 1},
{4, 1}, {3, 0}}, v=6, total output is {2, 2, 1, 5, 4, 3}
```
[3]

□ Декодирование

Для правильного декодирования получатель должен иметь возможность создать точно такой же словарь, как и у отправителя. Он может действовать лишь тогда, когда приходит новый символ. Пусть v — такой символ.

В соответствии с алгоритмом кодирования (алг. 6.10), поступление символа v означает, что к словарю был добавлен новый (скажем u-й) элемент. Указателем этой новой записи является v.

Символ источника этой новой записи не известен, поскольку он является корневым символом следующей строки, который пока еще не закодирован отправителем. Таким образом, пока в словарь можно добавить только пару $(v, ?)$.

Однако получатель теперь может заполнить отсутствующий символ в предыдущей записи словаря (по адресу $u-1$).

Далее получатель сможет декодировать всю строку символов источника, связанную с полученным символом.

Продемонстрируем описанный выше процесс для полученной последовательности из примера 6.4.

Пример 6.4 (часть 2)

Получатель выполняет такую же инициализацию, как и отправитель, т.е. $u = 3$, $v = 0$, *а словарь определяется как* $\{(0, \emptyset), (0, 0), (0, 1)\}$.

Он получает следующий список символов: $(2, 2, 1, 5, 4, 3)$.

Первый полученный символ — это $v = 2$.

Таким образом, новая запись словаря $(2, ?)$ *будет иметь адрес* $u = 3$. *Позицию, где поставлен вопрос, пока нельзя заполнить.*

Число 2 *в* $(2, ?)$ *указывает в словаре на запись с адресом* 2, *а это* $(0, 1)$. *Эта запись говорит, что последний символ предыдущей строки*

[3] Окончательно должно быть послано {2, 2, 1, 5, 4, 3, 6}. — *Прим. перев.*

был 1, *и что для поиска его предшественника нам следует перейти к записи словаря с адресом* 0. *Эта запись* $(0, \emptyset)$ *обозначает конец поиска. Декодированная строка — это* "1".

Новый словарь определяется как $\{(0, \emptyset), (0, 0), (0, 1), (2, ?)\}$.

Второй полученный символ — это $v = 2$.

Чтобы заполнить место в текущем словаре, помеченное вопросом, посмотрим на запись с адресом $v = 2$. *Это* $(0, 1)$. *Символ источника из нее следует поставить вместо вопроса. Следовательно, получится такой словарь:* $\{(0, \emptyset), (0, 0), (0, 1), (2, 1)\}$.

Кроме того, в словарь нужно добавить новую запись, а именно $(v, ?) = (2, ?)$ *по адресу* $u = 4$.

Число 2 *в этой новой записи* $(2, ?)$ *указывает в словаре на запись с адресом* 2, *а это* $(0, 1)$. *Эта запись говорит, что последний символ предыдущей строки был* 1, *и что для поиска его предшественника нам следует перейти к записи словаря с адресом* 0. *Эта запись* $(0, \emptyset)$ *обозначает конец поиска. Декодированная строка — это* "1".

Новый словарь определяется как $\{(0, \emptyset), (0, 0), (0, 1), (2, 1), (2, ?)\}$.

Третий полученный символ — это $v = 1$.

Чтобы заполнить место в текущем словаре, помеченное вопросом, посмотрим на запись с адресом $v = 1$. *Это* $(0, 0)$. *Символ источника из нее следует поставить вместо вопроса. Следовательно, получится такой словарь:* $\{(0, \emptyset), (0, 0), (0, 1), (2, 1), (2, 0)\}$.

Кроме того, в словарь нужно добавить новую запись, а именно $(v, ?) = (1, ?)$ *по адресу* $u = 5$.

Число 1 *в этой новой записи* $(1, ?)$ *указывает в словаре на запись с адресом* 1, *а это* $(0, 0)$. *Эта запись говорит, что последний символ предыдущей строки был* 0, *и что для поиска его предшественника нам следует перейти к записи словаря с адресом* 0. *Эта запись* $(0, \emptyset)$ *обозначает конец поиска. Декодированная строка — это* "0".

Новый словарь определяется как $\{(0, \emptyset), (0, 0), (0, 1), (2, 1), (2, 0), (1, ?)\}$.

Четвертый полученный символ — это $v = 5$.

Чтобы заполнить место в текущем словаре, помеченное вопросом, посмотрим на запись с адресом $v = 5$. *Это* $(1, ?)$. *Ее указатель* 1 *ссылается на другую запись в словаре, а именно на* $(0, 0)$. *Указатель* 0 *в этой записи означает, что мы находимся в начале строки. Символ источника из* $(0, 0)$ *означает, что* $? = 0$. *Следовательно, получится такой словарь:* $\{(0, \emptyset), (0, 0), (0, 1), (2, 1), (2, 0), (1, 0)\}$.

Кроме того, в словарь нужно добавить новую запись, а именно $(v, ?) = (5, ?)$ *по адресу* $u = 6$.

Число 5 *в этой новой записи* $(5, ?)$ *указывает в словаре на запись с адресом* 5, *а это* $(1, 0)$. *Эта запись говорит, что последний символ предыдущей строки был* 0, *и что для поиска его предшественника нам следует перейти к записи словаря с адресом* 1, *а это* $(0, 0)$. *Эта запись говорит, что предпоследний символ предыдущей строки был* 0, *и что*

для поиска его предшественника нам следует перейти к записи словаря с адресом 0. *Эта запись* $(0, \emptyset)$ *обозначает конец поиска. Декодированная строка — это* "00".

Новый словарь определяется как $\{(0, \emptyset), (0, 0), (0, 1), (2, 1), (2, 0), (1, 0), (5, ?)\}$.

Предлагаем читателю продолжить этот процесс.

6.4 Задачи

Задача 6.1. Декодируйте строку 01100111111111100011, полученную с помощью кода из примера 6.1.

Задача 6.2. Примените алгоритм Хаффмана к источнику открытых текстов $\mathcal{S}$, генерирующему символы a, b, c, d, e, f, g и h независимо друг от друга с вероятностями 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 и 1/128 соответственно. Каково среднее число битов, необходимых для кодирования одной буквы? Сравните это число с энтропией источника.

Задача 6.3^M. Повторите процесс, описанный в примере 6.3, для источника открытых текстов $\mathcal{S}$, генерирующего символы a, b, c, d, e, f, g и h независимо друг от друга с вероятностями 1/3, 1/4, 1/6, 1/12, 1/15, 1/20, 1/30 и 1/60 соответственно.

Задача 6.4. Примените вариант Уэллша алгоритма кодирования Лемпеля-Зива к последовательности 0000000000000000. Продемонстрируйте первые пять шагов процесса декодирования.

Глава 7

Криптография с публичными ключами

7.1 Теоретическая модель

7.1.1 Мотивировка и общая структура

С точки зрения современных систем связи традиционные криптографические системы имеют два существенных недостатка.

i) Проблема распределения ключей и управления ими.

Система связи с n участниками, использующими традиционную криптосистему для связи друг с другом, нуждается в $\binom{n}{2}$ ключах и $\binom{n}{2}$ безопасных каналах. Всякий раз, когда один из участников захочет изменить свои ключи или когда появляется новый участник системы, приходится генерировать и распределять $n-1$ (соответственно, n) новых ключей по столь же многим безопасным каналам.

ii) Проблема аутентификации.

В коммуникационных системах, управляемых компьютерами, необходим электронный эквивалент подписи. Традиционные криптосистемы не обеспечивают естественным образом эту потребность, особенно в случае конфликта между отправителем и получателем, когда невозможно решить, кто прав. Любое сообщение, отправленное одним из них, может быть также отправлено другим.

Эти недостатки побудили исследователей к поиску криптосистем иного рода.

У. Диффи и М. Хеллман опубликовали в [DifH76] пионерскую работу о *криптосистемах с публичными ключами*; см. рис. 7.1.

Каждый пользователь U криптосистемы создает пару согласованных алгоритмов P_U и S_U (или получает их из достойного доверия источника). Эти алгоритмы оперируют с элементами множеств, которые будут определены позже.

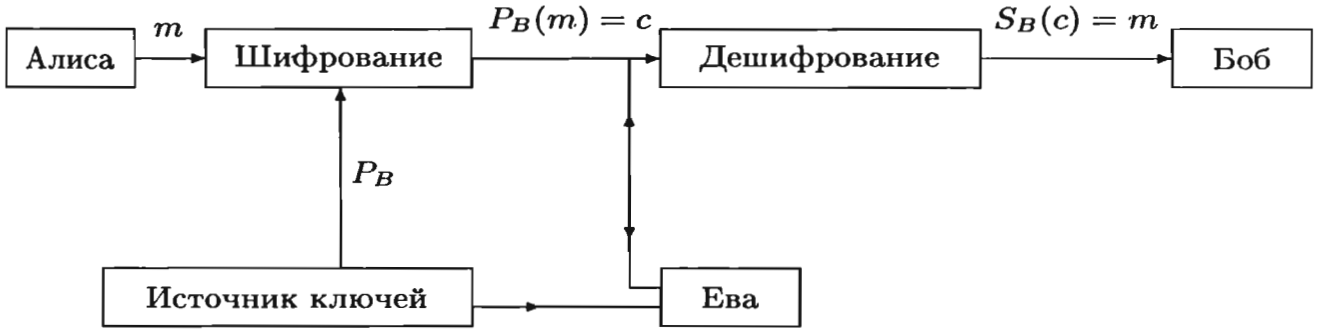


Рис. 7.1. Криптосистема с публичными ключами для шифрования.

Пользователь U должен объявить алгоритм P_U *публичным*, тогда как алгоритм S_U он должен держать в *секрете*. Эти алгоритмы, в зависимости от применения, должны удовлетворять некоторым из следующих условий:

ПК1. Алгоритмы P_U и S_U эффективны, т.е. не требуют слишком большого времени вычислений или большой памяти.

ПК2. $S_U(P_U(m)) = m$ для любого пользователя U и любого возможного сообщения m.

ПК3. Невозможно найти, исходя из P_U, алгоритм S_U^*, удовлетворяющий равенствам $S_U^*(P_U(m)) = m$ для всех m.

ПК4. $P_U(S_U(m)) = m$ для любого пользователя U и любого возможного сообщения m.

ПК5. Невозможно найти, исходя из P_U, алгоритм S_U^*, удовлетворяющий равенствам $P_U(S_U^*(m)) = m$ для всех m.

Условия ПК3 и ПК5 сформулированы неточно. Их точный смысл во многом зависит от применений и может меняться со временем.

7.1.2 Конфиденциальность

Допустим, что выполняются условия ПК1, ПК2 и ПК3.

Если Алиса хочет послать Бобу зашифрованное сообщение m, то она сначала отыскивает публичный алгоритм (шифрования) P_B Боба и шифрует m, применяя P_B. Таким образом, она посылает Бобу

$$c = P_B(m).$$

Боб из полученного шифртекста c восстанавливает m, применяя к c свой (секретный) алгоритм S_B. В самом деле,

$$S_B(c) = S_B(P_B(m)) \overset{\text{ПК2}}{=} m.$$

Чтобы система была практичной, должно выполняться условие ПК1. Для безопасности системы требуется условие ПК3, позволяющее публиковать алгоритмы шифрования P_U без угрозы конфиденциальности передаваемых сообщений.

Резюмируем схему шифрования в следующей таблице.

ПУБЛИЧНЫЕ	P_U ВСЕХ УЧАСТНИКОВ U
СЕКРЕТНЫЕ	S_U ДЛЯ ВСЕХ УЧАСТНИКОВ, КРОМЕ U
УСЛОВИЯ	ПК1, ПК2, ПК3
ШИФРОВАНИЕ m АЛИСОЙ	$P_B(m) = c$
ДЕШИФРОВАНИЕ c БОБОМ	$S_B(c) = m$

Таблица 7.1. Криптосистема с публичными ключами, используемая для шифрования.

Если пользователь U хочет изменить свой личный ключ, он просто генерирует новые согласованные алгоритмы P_U и S_U, удовлетворяющие условиям ПК1, ПК2 и ПК3, и объявляет P_U публичным. То же самое должен сделать новый пользователь, если захочет участвовать в данной системе связи.

Авторы публикации [DifH76] предложили использовать для шифрования одностороннюю функцию-ловушку. *Односторонняя функция* - это функция $f : A \longrightarrow B$ со следующими свойствами:

F1) $f(a)$ легко вычисляется для любого $a \in A$.

F2) вычислительно невозможно найти $f^{\leftarrow}(b)$ почти для всех $b \in B$.

Односторонняя функция-*ловушка* — это односторонняя функция с еще одним свойством

F3) $f^{\leftarrow}(b)$, $b \in B$, легко вычисляется, если известна некоторая дополнительная информация.

Свойство F1 делает такую функцию практичной в использовании, тогда как свойство F2 обеспечивает безопасность при использовании f в целях шифрования. Свойство F3 делает возможным дешифрование сообщений получателем.

В повседневной жизни в качестве односторонней функции можно использовать телефонную книгу; по заданному имени легко найти соответствующий телефонный номер, но не наоборот. Отыскание телефонного номера некой персоны равнозначно нахождению имени этой персоны. Это требует $\log_2 L$ операций, если L — число имен в телефонном справочнике. Нахождение же имени по заданному телефонному номеру означает просмотр всей книги, имя за именем. Сложность равна L. Свойство F2 базируется на экспоненциальной связи между $\log_2 L$ и L.

Односторонние функции f используются также для проверки аутентичности персоны, желающей получить доступ к некоторым данным. Каждый пользователь U имеет свой собственный ПИН-код (персональный идентификационный номер) x_U, но в центральном компьютере записано только имя U вместе со значением $y_U = f(x_U)$. Когда U захочет получить доступ, он должен ввести свое имя и x_U. Значение $f(x_U)$ будет вычислено и послано компьютеру. Если это значение совпадет с y_U, то пользователь U может получить доступ, а иначе нет. Преимущество этой системы состоит в том, что ПИН-коды x_U не надо хранить в компьютере. Поэтому никто, способный считывать память компьютера, не сможет определить ПИН-коды.

В главах 8, 9 и 12 мы обсудим различные предложения односторонних функций-ловушек, которые можно использовать для создания криптосистем с публичными ключами. В следующей главе мы встретимся с односторонней функцией, которая не имеет ловушки.

7.1.3 Цифровая подпись

Допустим, что выполняются условия ПК1, ПК4 и ПК5.

Если Алиса хочет подписать сообщение m, которое нужно послать Бобу, то она применяет к m свой собственный секретный алгоритм S_A и посылает

$$c = S_A(m).$$

Боб восстанавливает m из c, применяя к c публично известный алгоритм P_A. В самом деле,

$$P_A(c) = P_A(S_A(m)) \overset{\text{ПК4}}{=} m.$$

Однако возможно и обратное: любой способен найти такую пару (m, c), что c окажется подписью для m, т.е. такую, что $P_A(c) = m$: можно просто взять любое c и вычислить $m = P_A(c)$.

Поэтому Алиса должна позаботиться о том, чтобы случайно выбранное c имело ничтожную вероятность привести к полезному сообщению $P_A(c) = m$. Этого можно добиться довольно легко, предполагая некоторую структуру в каждом сообщении, например, начиная его с даты и времени.

Резюмируем объясненную выше систему подписи в следующей таблице.

ПУБЛИЧНЫЕ	P_U ВСЕХ УЧАСТНИКОВ U
СЕКРЕТНЫЕ	S_U ДЛЯ ВСЕХ УЧАСТНИКОВ, КРОМЕ U
УСЛОВИЯ	ПК1, ПК4, ПК5
ПОДПИСЫВАНИЕ m АЛИСОЙ	$S_A(m) = c$
ПРОВЕРКА c БОБОМ	$P_A(c) = m$

Таблица 7.2. Криптосистема с публичными ключами, используемая для подписывания сообщения.

Заметим, что любой другой пользователь тоже может проверить подпись Алисы, вычисляя $P_A(c)$, так что здесь нет никакой секретности.

7.1.4 Конфиденциальность и цифровая подпись

Допустим, что выполняются все условия ПК1–ПК5.

Если Алиса хочет послать Бобу сообщение m в зашифрованном виде и со своей подписью, то она комбинирует технику подразд. 7.1.2 и 7.1.3. Именно, используя собственный секретный алгоритм S_A и публичный алгоритм P_B Боба, она посылает

$$c = P_B(S_A(m)).$$

Боб восстанавливает m из c, применяя $P_A S_B$. В самом деле,

$$P_A(S_B((c)) = P_A(S_B(P_B(S_A(m)))) \overset{\text{ПК2}}{=} P_A(S_A(m)) \overset{\text{ПК4}}{=} m.$$

Хотя любой может прибегнуть к публичному P_B, только Боб может восстановить m из c, поскольку он один знает S_B.

Подписью Алисы Боб считает значение $S_B(c)$, т.е. $S_B(P_B(S_A(m)))$, равное $S_A(m)$.

Резюмируем все это в следующей таблице.

ПУБЛИЧНЫЕ	P_U ВСЕХ УЧАСТНИКОВ U
СЕКРЕТНЫЕ	S_U ДЛЯ ВСЕХ УЧАСТНИКОВ, КРОМЕ U
УСЛОВИЯ	ПК1, ПК2, ПК3, ПК4, ПК5
АЛИСА ПОСЫЛАЕТ	$P_B(S_A(m)) = c$
БОБ ВЫЧИСЛЯЕТ	$P_A(S_B(c)) = m$
БОБ СОХРАНЯЕТ	$S_B(c) = S_A(m)$

Таблица 7.3. Криптосистема с публичными ключами, используемая для шифрования и подписывания.

7.2 Задачи

Задача 7.1. В коммуникационной сети каждый пользователь U имеет свои собственные публичный алгоритм шифрования P_U и секретный алгоритм дешифрования S_U. Сообщение m от пользователя A (Алисы) пользователю B (Бобу) всегда посылается в формате (c, A), где $c = P_B(m)$. Имя отправителя в этом сообщении говорит Бобу, от кого пришло сообщение.

Боб восстанавливает m из (c, A), вычисляя $S_B(c) = S_B(P_B(m)) = m$ (см. ПК2), но Боб, кроме того, автоматически посылает Алисе пару $(P_A(m), B)$ (заметим, что $(P_A(m), B)$ имеет тот же формат, что и $(P_B(m), A)$). Таким образом, Алиса узнает, что ее сообщение правильно получено Бобом.

а) Покажите, как третий пользователь сети E (Ева) может восстановить сообщение m, посланное Бобу Алисой. Можно предполагать, что Ева способна перехватывать все сообщения, передаваемые по сети, а любой C может передавать свои собственные тексты при условии, что они имеют правильный формат.

б) Покажите, что связь по этой сети по-прежнему остается небезопасной, если протокол таков, что Алиса посылает Бобу $P_B((P_B(m), A))$, а Боб автоматически посылает обратно Алисе $P_A((P_A(m), B))$.

Глава 8

Системы, основанные на дискретных логарифмах

8.1 Система дискретных логарифмов

8.1.1 Проблема дискретных логарифмов

Диффи и Хеллман в [DifH76] предложили систему распределения публичных ключей, которая базируется на несомненной трудности вычисления логарифмов над конечным полем GF(p), p простое, которое часто обозначается через $\mathbb{F}_p$ или $\mathbb{Z}_p$. Читатель, не знакомый с теорией конечных полей, отсылается к приложению В.

Пусть α — примитивный элемент (или образующий) поля GF(p). Тогда каждый ненулевой элемент c из GF(p) может быть записан в виде

$$c = \alpha^m, \tag{8.1}$$

где m единственно по модулю $p-1$.

Пример 8.1

В GF(7) *элемент* $\alpha = 3$ *примитивен, что легко проверить вычислениями:* $3^2 \equiv 2 \pmod 7$, $3^3 \equiv 6 \pmod 7$, $3^4 \equiv 4 \pmod 7$, $3^5 \equiv 5 \pmod 7$, $3^6 \equiv 1 \pmod 7$. *Это можно сделать одновременно:*

```
Mod[3 ^ {1, 2, 3, 4, 5, 6}, 7]
```

```
{3, 2, 6, 4, 5, 1}
```

Пример 8.2

В GF(197) *элемент* $\alpha = 2$ *примитивен. Такой элемент можно найти с помощью функции* PowerList *пакета "Mathematica" (для чего сначала нужно инициализировать пакет Algebra'FiniteFields'). Эта функция находит примитивный элемент в* $\mathbb{F}_p$ *и порождает все его степени (начиная с* 0*-й). Второй элемент получаемого списка — сам примитивный элемент.*

```
<< Algebra'FiniteFields'
```

```
p = 197;
PowerList[GF[p, 1]][[2]]
```

```
|| {2}
```

Проверить, что 2 *— примитивный элемент по модулю* 197, *гораздо легче. Мультипликативная группа* $\mathbb{Z}_{197}^*$ *имеет порядок* 196, *так что порядок каждого элемента делит* 196 *(см. теорему B.5). Функция* FactorInteger *помогает найти все различные простые делители числа* 196.

```
FactorInteger[196]
```

```
|| {{2, 2}, {7, 2}}
```

Теперь из

```
PowerMod[2, 196 / 7, 197] == 1
PowerMod[2, 196 / 2, 197] == 1
```

```
|| False

|| False
```

следует, что порядок элемента 2 *по модулю* 197 *не делит* 196/2 *и* 196/7, *так что этот порядок должен быть равен* 196.

Если m задано, то c можно вычислить по формуле (8.1) с помощью $2\cdot\lceil\log_2 p\rceil$ умножений (см. [Knut81] или [Knut69], разд. 4.6.3). Это можно реализовать, создав таблицу α, α^2, α^{2^2}, α^{2^3},..., $\alpha^{2^{\lceil\log_2 p\rceil-1}}$, (каждая степень — квадрат предыдущей) и перемножив те элементы этой таблицы, чьи показатели при сложении дают m. Для этой цели можно использовать двоичное представление m.

Пример 8.3

Возьмем $m = 171$. *Его двоичным представлением служит* 10101011, *как показывает функция* IntegerDigits *пакета "Mathematica":*

```
IntegerDigits[171, 2]
```

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

Получается $\alpha^{171} = \alpha^{128}\cdot\alpha^{32}\cdot\alpha^{8}\cdot\alpha^{2}\cdot\alpha$. *Это вычисление можно сделать "на лету". Самая левая единица в двоичном представлении* m *заменяется на* α. *Каждый последующий символ влечет возведение в квадрат предыдущего результата, но если этот символ есть* 1, *то полученный квадрат дополнительно умножается на* α.

```
Clear[a];
(((((((a^2)^2 a)^2)^2 a)^2)^2 a)^2 a
```

```
|| a^171
```

Если модулярное возведение в одну и ту же степень нужно выполнять много раз, например, в smart-карте, то имеются способы делать это с меньшим числом умножений.

Определение 8.1
Аддитивной цепочкой для натурального m называется последовательность натуральных чисел $a_1 = 1 < a_2 ... < a_{\ell-1} < a_\ell = m$ с тем свойством, что каждое a_k, $2 \le k \le \ell$, есть сумма двух (не обязательно различных) предыдущих чисел. Число ℓ называют *длиной* цепочки.

Способ использования аддитивных цепочек для (модулярного) возведения в степень ясен. Если $a_k = a_i + a_j$, то $\alpha^{a_k} = \alpha^{a_i} \cdot \alpha^{a_j}$. Следовательно, $\alpha^m = \alpha^{a_\ell}$ можно вычислять рекурсивно. Но, вообще говоря, не очевидно, как находить кратчайшую аддитивную цепочку для натурального m. См. [Knut81] или [Knut69], разд. 4.6.3, и [Bos92], гл. 4.

Пример 8.4

Последовательность $1, 2, 3, 6, 12, 15$ *является аддитивной цепочкой для* $m = 15$. *Заметим, что вычисление* α^{15} *с этой аддитивной цепочкой включает* 5 *умножений, а с помощью описанного ранее бинарного метода* 6 *умножений.*

Функция *PowerMod* пакета "Mathematica" реализует быстрый способ модулярного возведения в степень.

```
a = 2; m = 171111111; p = 197888888;
PowerMod[a, m, p]
```

```
55895160
```

Обратная задача нахождения m, удовлетворяющего (8.1), по данному c не столь легка. Она называется *проблемой дискретных логарифмов*, потому что в $\mathbb{Z}_p$ показатель m можно записать как $m = \log_\alpha c$. В [Knut73], с. 9, 575–576 (с. 22, 681 русского перевода) можно найти алгоритм, решающий проблему логарифмов. Он требует грубо $c_1\sqrt{p}$ операций и $c_2\sqrt{p}$ битов памяти (где c_1 и c_2 — некоторые константы). В теореме 8.1 будет дан более точный анализ этого алгоритма. Записывая $t = \log_2 p$ (и забывая о константах), получаем следующую экспоненциальную связь между возведением в степень и взятием логарифмов.

Возведение в степень	t
Взятие логарифмов	$2^{t/2}$

Таблица 8.1. Экспоненциальное расхождение между возведением в степень и взятием логарифмов.

8.1.2 Система Диффи-Хеллмана обмена ключами

Теперь мы опишем, как можно использовать расхождение во времени вычисления между возведением в степень и взятием логарифмов, отраженное в табл. 8.1, чтобы выполнить *протокол обмена ключами* в духе криптографии с публичными ключами. Такой протокол является безопасным методом согласования общего секретного ключа двумя сторонами, которые до того не разделяли общий ключ.

Формирование системы:

1) Все участники разделяют в качестве системных параметров простое число p и примитивный элемент α в $\mathrm{GF}(p)$.

2) Каждый участник P выбирает случайное целое число m_P, $1 < m_P \leq p-2$, вычисляет $c_P = \alpha^{m_P}$ и помещает его в книгу публичных ключей. Участник P держит m_P в секрете.

Использование системы:

Допустим теперь, что Алиса (для краткости A) и Боб (B) хотят общаться друг с другом, используя традиционную криптосистему, но у них нет безопасного канала для обмена ключом. С помощью книги публичных ключей они могут согласовать общий секретный ключ

$$k_{A,B} = \alpha^{m_A m_B}.$$

Алиса может вычислить $k_{A,B}$, возводя публично известное c_B в степень m_A, известную лишь ей одной. В самом деле,

$$(c_B)^{m_A} = (\alpha^{m_B})^{m_A} = \alpha^{m_A m_B} = k_{A,B}.$$

Аналогично Боб находит $k_{A,B}$, вычисляя $(c_A)^{m_B}$.

Если кто-то другой (скажем, Ева) способен вычислить m_A, исходя из c_A (или m_B, исходя из c_B), то этот кто-то может вычислить ключ $k_{A,B}$ в точности так же, как это сделали Алиса или Боб. При достаточно большом p время вычислений, необходимое для решения этой проблемы логарифмов, становится недоступно большим. И, видимо, не существует иного пути нахождения $k_{A,B}$, исходя из c_A и c_B. Диффи и Хеллман предлагают брать p около 100 битов длиной.

Нет никаких очевидных резонов ограничивать размер конечного поля простым числом. Впредь размер поля может быть произвольной степенью $q = p^e$ простого числа (см. теорему В.16 или теорему В.20). В [Lune87], гл. XIII, описаны эффективные алгоритмы нахождения примитивных элементов в конечных полях. См. также задачи В.6 и В.10.

Резюмируем систему распределения ключей в табл. 8.2.

СИСТЕМНЫЕ ПАРАМЕТРЫ	РАЗМЕР ПОЛЯ q ПРИМИТИВНЫЙ ЭЛЕМЕНТ α
СЕКРЕТНЫЙ КЛЮЧ P	m_P
ПУБЛИЧНЫЙ КЛЮЧ P	$c_P = \alpha^{m_P}$
ОБЩИЙ КЛЮЧ A И B	$k_{A,B} = \alpha^{m_A m_B}$
АЛИСА ВЫЧИСЛЯЕТ	$(c_B)^{m_A}$
БОБ ВЫЧИСЛЯЕТ	$(c_A)^{m_B}$

Таблица 8.2. Система Диффи-Хеллмана обмена ключами.

Пример 8.5 (часть 1)

Пусть $p = 197$ *и* $\alpha = 2$. *Алиса случайным образом выбирает секретный показатель* $m_A = 56$, *а Боб аналогично выбирает секретный показатель* $m_B = 111$. *Они вычисляют свои публичные ключи с помощью функции* PowerMod:

```
cA = PowerMod[2, 56, 197]
cB = PowerMod[2, 111, 197]
```

```
178
```

```
82
```

Алиса может вычислить общий с Бобом ключ, возводя публично известное c_B *в степень* m_A, *известную только ей, и получая*

```
PowerMod[82, 56, 197]
```

```
114
```

Боб получает тот же самый общий ключ, возводя c_A *в степень* m_B. *В самом деле,*

```
PowerMod[178, 111, 197]
```

```
114
```

8.2 Другие системы, основанные на дискретных логарифмах

8.2.1 Криптосистемы с публичными ключами Эль-Гамаля

В [ElGa88] описаны две системы с публичными ключами, основанные на проблеме дискретных логарифмов. Одна из них используется для шифрования, другая — как схема подписи. В обеих системах передаваемый текст длиннее исходного открытого.

□ Формирование системы

Все участники разделяют в качестве системных параметров простое число p и образующий (примитивный элемент) α мультипликативной группы $\mathbb{Z}_p^*$. Обобщение на произвольные конечные поля очевидно и будет опущено.

Довольно часто встречается вариация: рассматривают $\mathbb{Z}_q^*$ с простым q, а вместо примитивного элемента берут элемент $\alpha \in \mathbb{Z}_q^*$ большого простого порядка, скажем, p. Отметим, что по теореме B.5 число p должно делить $q-1$.

Каждый участник P случайным образом выбирает целое число m_P, $1 \le m_P \le p-2$, вычисляет $c_P = \alpha^{m_P}$, объявляет c_P публичным, а m_P держит в секрете.

Как вариация, каждый участник вместо общесистемных параметров может выбрать свои собственные конечное поле и примитивный элемент, но, видимо, большого резона в том нет.

□ Секретная система Эль-Гамаля

Шифрование сообщения для Боба.

Предположим, что Алиса хочет послать Бобу секретное сообщение. Сообщение представлено целым числом u из $\{0, 1, \ldots, p-1\}$. Алиса выбирает случайное секретное r и вычисляет $R = \alpha^r \bmod p$. Затем Алиса вычисляет $S = u \cdot c_B^r \bmod p$ и посылает Бобу пару (R, S).

Дешифрование сообщения Бобом.

Боб получает пару (R, S) и, используя свое секретное m_B, легко восстанавливает сообщение u путем следующего вычисления:

$$S/R^{m_B} = u \cdot c_B^r/\alpha^{r \cdot m_B} = u \cdot \alpha^{r \cdot m_B}/\alpha^{r \cdot m_B} = u.$$

Пример 8.5 (часть 2)

Продолжаем пример 8.5. *Наши публичные параметры:* $p = 197$, $\alpha = 2$, $c_B = 82$. *Число* $m_B = 111$ *известно только Бобу.*

Предположим, что Алиса хочет зашифровать для Боба сообщение $u = 123$. *Пусть* $r = 191$ — *случайное целое число, выбранное Алисой (оно простое). Алиса посылает пару* (R, S), *вычисленную так:*

```
p = 97; a = 2; cB = 82;
r = Random[Integer, 0, p-2]
u = 123;
R = PowerMod[a, r, 197]
S = Mod[PowerMod[cB, r, 197] * u, p]
```

```
191
117
175
```

При дешифровании Боб вычисляет $S/R_B^m \bmod p$, используя свое секретное m_B и функции *Mod*, *PowerMod* пакета "Mathematica". Заметим, что PowerMod$[a, -1, p]$ вычисляет мультипликативный обратный элемент к a по модулю p (см. подразд. А.3.3).

```
mB = 111;
Mod[S * PowerMod[PowerMod[R, mB, p], -1, p], p]
```

```
123
```

Перехватчик не может определить r, исходя из R, так как мы предполагаем, что проблема логарифмов не поддается решению. В силу этого перехватчик не способен выделить c_B^r из S (чтобы получить секретное u).

□ Схема подписи Эль-Гамаля

Подписывание сообщения Алисой.

Предположим, что Алиса хочет послать Бобу подписанное сообщение. Сообщение снова представлено числом u из $\{0, 1, \ldots, p-2\}$[1].

Алиса выбирает случайное целое r, взаимно простое с $p-1$, и вычисляет $R = \alpha^r \bmod p$. Затем Алиса использует свой секретный показатель m_A для вычисления числа S, удовлетворяющего сравнению

$$u \equiv m_A \cdot R + r \cdot S \pmod{p-1}. \tag{8.2}$$

Чтобы эффективно найти S, можно использовать расширенную версию алгоритма Эвклида.

Алиса посылает Бобу тройку (u, R, S), где пара (R, S) служит подписью к сообщению u.

Проверка подписи Бобом.

Боб получает подпись (R, S) вместе с сообщением u и проверяет эту подпись, убеждаясь, что

$$\alpha^u = (c_A)^R \cdot R^S \pmod{p}.$$

Это сравнение должно выполняться, так как в силу соотношения (8.2)

$$\alpha^u \equiv \alpha^{m_A \cdot R} \alpha^{r \cdot S} \equiv (\alpha^{m_A})^R \cdot (\alpha^r)^S \equiv (c_A)^R \cdot R^S \pmod{p}.$$

Пример 8.5 (часть 3)

Продолжаем пример 8.5 *с публичными параметрами* $p = 197$, $\alpha = 2$ *и* $c_A = 178$. *Число* $m_A = 56$ *известно только Алисе. Предположим, что Алиса хочет подписать для Боба сообщение* $u = 123$. *Пусть* $r = 97$ · *случайное число, выбранное Алисой (оно простое). Алиса вычисляет*

[1] Это может быть хэш-значение какого-нибудь длинного сообщения. — *Прим. ред.*

```
p = 197; a = 2; mA = 56;
r = 97; u = 123; S =.;
R = PowerMod[a, r, 197]
S /. Solve[r * S == u - mA * R, Modulus == p-1, S][[1]]
```

```
97
```

```
98
```

```
171
```

чтобы найти подпись $(R, S) = (98, 171)$, *которую она добавляет к своему сообщению* u.

Боб проверяет эту подпись, убеждаясь, что $\alpha^u \equiv (c_A)^R \cdot R^S \pmod{p}$:

```
cA = 178; R = 98; S = 171;
PowerMod[a, u, p] ==
    Mod[PowerMod[cA, R, p] * PowerMod[R, S, p], p]
```

```
True
```

8.2.2 Дальнейшие вариации

Подпись к сообщению u в схеме Эль-Гамаля состоит из двух частей: R, равного α^r со случайным r, и S, являющего решением сравнения $u \equiv m_A \cdot R + r \cdot S \pmod{p-1}$ (см. (8.2)). Конечно, можно варьировать это так называемое *уравнение подписи*. В следующих трех схемах именно это и делается. Читатель, который хочет узнать о них больше, чем представлено ниже, отсылается к [MeOoV96] и [Schne96].

□ Стандарт цифровой подписи

В американском *стандарте цифровой подписи DSS* (Digital Signature Standard; см. [FIPS94]) уравнение подписи имеет вид

$$r \cdot S \equiv u + m_A \cdot R \pmod{p-1}.$$

Эта система разработана Агенством национальной безопасности (АНБ, по-английски NSA) и принята в качестве стандарта Национальным институтом стандартов и технологии (NIST) США.

DSS добавляет к концу документа две последовательности, по 160 битов каждая, в качестве гарантии его аутентичности и целостности. С этой целью DSS сначала сжимает документ до 160-битовой последовательности с помощью криптографически безопасной хэш-функции (см. разд. 13.2), называемой SHA (*Secure Hash Algorithm*; см. [MeOoV96] и [Schne96]).

Для установки системы выбирают следующие общие параметры:

i) простое число q, двоичное представление которого имеет длину, кратную 64 и лежащую между 512 и 1024;

ii) простой делитель p числа $q-1$, имеющий длину 160 битов;

iii) значение $g = h^{(q-1)/p} \bmod q$, большее 1, где $h < q-1$.

Так как по теореме Ферма (A.15) $g^p \equiv h^{q-1} \equiv 1 \pmod{q}$, мультипликативный порядок элемента g делит p. Но поскольку p простое, этот порядок равен p (см. также теорему B.5).

Каждый пользователь U выбирает секретный показатель m_U, вычисляет $c_U = g^{m_U} \bmod q$ и делает c_U публичным.

Когда Алиса хочет подписать файл M, она сначала вычисляет с помощью SHA хэш-значение $h(M)$ длиной 160 битов. Затем она выбирает случайное число $r < p$ и добавляет в качестве подписи к M числа R и S длиной по 160 битов, определяемые равенствами

$$\begin{aligned} R &= (g^r \bmod q) \bmod p, \\ S \cdot r &= (h(M) + m_A \cdot R) \bmod p. \end{aligned}$$

Получатель проверяет аутентичность и целостность полученного сообщения M, вычисляя

$$\begin{aligned} w &= S^{-1} \bmod p, \\ x &= h(M) \cdot w \bmod p, \\ y &= R \cdot w \bmod p, \\ U &= (g^x \cdot c_A^y \bmod q) \bmod p. \end{aligned}$$

Если $R = U$, документ считается подлинным и пришедшим от Алисы. Простой подстановкой можно проверить, что равенство $R = U$ действительно должно выполняться. Назначение случайного числа r — спрятать секретный ключ Алисы.

□ Схема подписи Шнорра

В этой схеме ([Schno90]) уравнение подписи (см. (8.2)) имеет вид[2]

$$S = m_A \cdot R + r \bmod (p-1).$$

□ Схема подписи Ниберга–Руппеля

Эта схема ([NybR93]) слегка отличается от других. Здесь R определяется равенством

$$R = u \cdot \alpha^r \bmod p,$$

[2] В этой и следующей схемах $p-1$ может заменяться некоторым большим $q|p-1$. Параметры систем в общем случае определяются точно так же, как в DSS; только модуль $p-1$ нужно всюду заменить модулем q. — *Прим. ред.*

где u — сообщение, а r — случайное число. Уравнение подписи (см.(8.2)) имеет вид

$$S = m_A \cdot R - r \bmod (p-1).$$

В схеме Ниберга–Руппеля сообщение u можно восстановить прямо из R и S, так как

$$u \equiv R \cdot \alpha^{-r} \equiv R \cdot \alpha^{S-m_A \cdot R} \equiv R \cdot \alpha^S / (\alpha_A^m)^R \equiv R \cdot \alpha^S / (c_A)^R \pmod{p}.$$

Если u не является хэш-значением другого, более длинного файла, то эта особенность будет преимуществом, поскольку достаточно послать только R и S[3].

8.3 Как брать дискретные логарифмы

Когда нужно взять логарифм в GF(q), наиболее очевидный путь сокращения работы — разложить $q-1$ на множители, являющиеся степенями простых чисел, вычислить логарифм для каждого такого множителя и воспользоваться китайской теоремой об остатках (теорема А.19). Этот метод будет продемонстрирован в подразд. 8.3.1.

Как уже было сказано раньше, системы, основанные на дискретных логарифмах, часто работают в рамках какой-либо мультипликативной подгруппы из GF(q). Это обобщение не влияет на методы, обсуждаемые в данном разделе.

8.3.1 Алгоритм Похлига–Хеллмана

Похлиг и Хеллман в [PohH78] показали, что дискретные логарифмы могут вычисляться много быстрее, чем за $\sqrt{q}$ операций, если $q-1$ имеет только малые простые делители. Сначала мы продемонстрируем их метод в двух частных случаях.

□ **Частный случай:** $q - 1 = 2^n$

Примерами простых чисел, которые являются степенью двойки плюс единица, служат[4] $q = 17, q = 257$ и $q = 2^{16} + 1$.

```
n = 16; Prime[2^n + 1]
```

```
True
```

Итак, пусть α — примитивный элемент в конечном поле GF(q). Задача состоит в том, чтобы найти $m, 0 \le m \le q-2$, удовлетворяющее равенству (8.1) для заданного значения c.

[3] См. также российский стандарт цифровой подписи *[ГОСТП94], близкий к рассмотренным здесь системам, особенно к DSS. — *Прим. ред.*

[4] Если еще добавить числа 3 и 5, то список известных простых указанного вида (их называют простыми числами Ферма) будет исчерпан. — *Прим. ред.*

Пусть $m_0, m_1, \ldots, m_{n-1}$ — двоичное представление неизвестного m, т.е.

$$m = m_0 + m_1 2 + \ldots + m_{n-1} 2^{n-1}, m_i \in \{0,1\}, 0 \le i \le n-1.$$

Разумеется, достаточно вычислить неизвестные m_i. Так как α примитивный элемент в GF(q), мы знаем (см. теорему B.21), что $\alpha^{q-1} = 1$ и $\alpha^i \neq 1$, если $0 < i < q-1$. Отсюда следует также, что $\alpha^{(q-1)/2} = -1$, потому что квадрат элемента $\alpha^{(q-1)/2}$ равен 1, причем $\alpha^{(q-1)/2} \neq 1$. (Мы пользуемся тем, что по теореме B.15 корнями квадратного уравнения $x^2 = 1$ служат лишь ± 1.) Следовательно,

$$c^{(q-1)/2} = (\alpha^m)^{(q-1)/2} = \alpha^{(m_0+m_1 2+\ldots+m_{n-1}2^{n-1})(q-1)/2} =$$

$$\overset{\alpha \text{ прим.}}{=} \alpha^{m_0(q-1)/2} = \begin{cases} +1, & \text{если } m_0 = 0, \\ -1, & \text{если } m_0 = 1. \end{cases}$$

Поэтому вычисление $c^{(q-1)/2}$, которое, как мы видели в подразд. 8.1.1, требует самое большее $2\lceil \log_2 q \rceil$ умножений, дает m_0.

Вычислим $c_1 = c\alpha^{-m_0}$. Теперь m_1 можно определить так же, как и выше:

$$c_1^{(q-1)/4} = \alpha^{(m_1 2 + m_2 2^2 + \ldots + m_{n-1}2^{n-1})(q-1)/4} =$$

$$= \alpha^{m_1(q-1)/2} = \begin{cases} +1, & \text{если } m_1 = 0, \\ -1, & \text{если } m_1 = 1. \end{cases}$$

Далее вычисляем $c_2 = c_1\alpha^{-2m_1} = c\alpha^{-(m_0+m_1 2)}$ и определяем m_2 из $c_2^{(q-1)/8}$. Повторяем этот процесс, пока не будет определено m_{n-1} (и, значит, m).

Описанный выше алгоритм находит степень m, исходя из числа c, используя самое большее

$$n \cdot (2 \cdot \lceil \log_2 q \rceil + 2) \approx 2 \cdot (\log_2 q)^2 \approx 2n^2$$

операций; слагаемое "2" в левой части оценивает вычисление c_i (на i-м шаге $\alpha^{-2^{i-1}}$ нужно возвести в квадрат и результат, возможно, умножить на c_{i-1}).

Сравнение с табл. 8.1 показывает, что в данном случае (когда $q = 2^n + 1$) расхождение между вычислительной сложностью использования схемы Диффи–Хеллмана (одно возведение в степень, включающее максимум $2n$ умножений) и ее взлома ($\approx 2n^2$ умножений) квадратично, что не достаточно для гарантии безопасности системы.

Замечание. В случае $q-1 = s \cdot 2^t$, где s нечетно, t наименее значащих битов могут быть найдены точно таким же образом.

Пример 8.6

Рассмотрим уравнение $3^m \equiv 7 \pmod{17}$. *Таким образом,* $q = 17$, $\alpha = 3, c = 7$. *Заметим, что* $\alpha^{-1} = 6$. *Записывая* $m = m_0 + 2m_1 + 4m_2 + 8m_3$, *находим* m_0 *при вычислении* $c^{(q-1)/2} \bmod q$.

```
PowerMod[7, 8, 17]
```

|| 16

Так как это -1, *мы узнаем, что* $m_0 = 1$. *Вычисляем* $c_1 \equiv c/3 \equiv 6 \cdot c \equiv 8 \pmod{17}$. *Теперь можно найти* m_1, *исходя из* $c_1^{(q-1)/4} \bmod q$:

```
PowerMod[8, 4, 17]
```

|| 16

Снова получаем -1, *так что* $m_1 = 1$. *Вычисляем* $c_2 \equiv c/3^2 \equiv 6^2 \cdot c_1 \equiv 16 \pmod{17}$. *Теперь можно найти* m_2 *из* $c_2^{(q-1)/8} \bmod q$:

```
PowerMod[16, 2, 17]
```

|| 1

Так как результат равен 1, *имеем* $m_2 = 0$. *Значит,* $c_3 = c_2$ *и* m_3 *можно найти из* $c_3^{(q-1)/16} \bmod q$:

```
PowerMod[16, 1, 17]
```

|| 16

Теперь мы имеем $m_3 = 1$ *и поэтому* $m = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 = 11$. *Это легко проверить:*

```
PowerMod[3, 11, 17]
```

□ Общий случай: $q - 1$ имеет только малые простые делители

Пусть $q - 1 = \prod_{i=1}^{k} p_i^{n_i}$, где p_i — различные простые, а показатели n_i строго положительны (см. основную теорему теории чисел, теорема А.6). Мы предполагаем, что все p_i малы. Позже мы уточним, что под этим подразумевается.

Вместо прямого нахождения m из (8.1) мы будем определять

$$m^{(i)} \equiv m \pmod{p_i^{n_i}}, \quad 1 \le i \le k. \tag{8.3}$$

С помощью китайской теоремы об остатках (теорема А.19) из чисел $m^{(i)}$ можно эффективно вычислить m.

Чтобы определить $m^{(1)}$ (другие $m^{(i)}$ можно найти тем же путем), мы записываем его в p_1-ичном представлении. Удобства ради мы опускаем все верхние и нижние индексы, ссылающиеся на случай $i = 1$:

$$m^{(1)} = m_0 + m_1 p + \ldots + m_{n-1} p^{n-1}, m_i \in \{0, 1, \ldots, p-1\}, 0 \le i \le n-1.$$

Подобно частному случаю ($k = 1, p = 2$) мы будем находить каждый из коэффициентов m_i единственным возведением в степень.

Коэффициент m_0 может быть найден вычислением $c^{(q-1)/p}$. Из теоремы B.21 следует равенство $(c^{(q-1)/p})^p = 1$, означающее, что $c^{(q-1)/p}$ – корень степени p из единицы. Определим примитивный корень ω степени p из единицы как $\omega = \alpha^{(q-1)/p}$ и построим *таблицу*: $1, \omega, \omega^2, \ldots, \omega^{p-1}$. Тогда, поскольку $m \equiv m^{(1)} \pmod{p^n}$ и $m^{(1)} \equiv m_0 \pmod{p}$, мы имеем:

$$c^{(q-1)/p} = (\alpha^m)^{(q-1)/p} = \alpha^{m(q-1)/p} = \alpha^{m^{(1)}(q-1)/p} = \alpha^{m_0(q-1)/p} = \omega^{m_0}.$$

Поэтому простой поиск $c^{(q-1)/p}$ в таблице дает m_0.

Чтобы определить m_1, сначала вычисляем $c_1 = c \cdot \alpha^{-m_0}$, затем вычисляем $c_1^{(q-1)/p^2}$ и т.д., пока не определим $m^{(1)}$. Аналогичные вычисления должны быть проделаны для определения других $m^{(i)}$.

В этом алгоритме нужно создавать таблицы степеней примитивных корней степени p из единицы для всех простых делителей числа $q-1$. Эти делители должны быть достаточно малы, чтобы можно было записать таблицы.

Каждый раз, когда мы хотим взять логарифм, алгоритм должен произвести $\sum_{i=1}^{k} n_i$ возведений в степень; поэтому алгоритм использует

$$\sum_{i=1}^{k} 2 \cdot \lceil \log_2 q \rceil \cdot n_i \approx 2 \cdot \log_2 q \cdot (\sum_{i=1}^{k} n_i) \leq 2(\log_2 q)^2$$

операций, если мы забудем о членах низшего порядка. Мы снова получаем квадратичное расхождение между использованием системы обмена ключами Диффи–Хеллмана и ее взломом.

□ Пример применения алгоритма Похлига–Хеллмана

Пример 8.7

Рассмотрим уравнение (8.1) *с* $q = 8101$ *и примитивным элементом* $\alpha = 6$. *Заметим, что число* q *просто, значит,* $\mathrm{GF}(q) = \mathbb{Z}_{8101}$.

Предварительные вычисления.

Прежде всего разложим $q-1$ *и вычислим мультипликативный обратный элемент к* 6 *по модулю* 8101 *с помощью функций* FactorInteger *и* PowerMod *пакета "Mathematica".*

```
q = 8101; a = 6;
FactorInteger[q - 1]
x = PowerMod[a, -1, q]
```

```
|| {{2, 2}, {3, 4}, {5, 2}}
```

```
|| 6751
```

Таким образом, $q-1 = 2^2 \cdot 3^4 \cdot 5^2$ *и* $\alpha^{-1} = 6751$. *Теперь мы вновь используем функцию* PowerMod *для вычисления примитивных корней степеней* 2, 3 *и* 5 *из единицы:* $\omega_1 = 6^{(8101-1)/2} = 6^{4050}$; $\omega_2 = 6^{(8101-1)/3} = 6^{2700}$ *и* $\omega_3 = 6^{(8101-1)/5} = 6^{1620}$:

```
q = 8101; a = 6;
Om1 = PowerMod[a, (q - 1) / 2, q];
Om2 = PowerMod[a, (q - 1) / 3, q];
Om3 = PowerMod[a, (q - 1) / 5, q];
```

```
8100
```

```
5883
```

```
3547
```

Итак, $\omega_1 = 8100$, $\omega_2 = 5883$ *и* $\omega_3 = 3547$. *С помощью функции* Table *создаем следующие три таблицы:*

```
q = 8101; a = 6;
Om1 = PowerMod[a, (q - 1) / 2, q];
Om2 = PowerMod[a, (q - 1) / 3, q];
Om3 = PowerMod[a, (q - 1) / 5, q];
Table[PowerMod[Om1, i, q], {i, 0, 1}]
Table[PowerMod[Om2, i, q], {i, 0, 2}]
Table[PowerMod[Om3, i, q], {i, 0, 4}]
```

```
{1, 8100}
```

```
{1, 5883, 2217}
```

```
{1, 3547, 356, 7077, 5221}
```

Следовательно, мы имеем таблицы:

$p_1 = 2$	i	0	1
	$(\omega_1)^i$	1	8100

$p_2 = 3$	i	0	1	2
	$(\omega_2)^i$	1	5883	2217

$p_3 = 5$	i	0	1	2	3	4
	$(\omega_3)^i$	1	3547	356	7077	5221

Предварительная работа для китайской теоремы об остатках состоит в решении следующих трех систем линейных сравнений:

$$\begin{cases} u \equiv 1 \pmod 4, \\ u \equiv 0 \pmod{81}, \\ u \equiv 0 \pmod{25}. \end{cases}$$

$$\begin{cases} v \equiv 0 \pmod 4, \\ v \equiv 1 \pmod{81}, \\ v \equiv 0 \pmod{25}. \end{cases}$$

$$\begin{cases} w \equiv 0 \pmod 4, \\ w \equiv 0 \pmod{81}, \\ w \equiv 1 \pmod{25}. \end{cases}$$

Эти три системы могут быть решены с помощью функции ChineseRemainderTheorem *пакета "Mathematica", для чего сначала нужно загрузить пакет NumberTheory'NumberTheoryFunctions'.*

```
<< NumberTheory`NumberTheoryFunctions`
```

```
ChineseRemainderTheorem[{1, 0, 0}, {4, 81, 25}]
ChineseRemainderTheorem[{0, 1, 0}, {4, 81, 25}]
ChineseRemainderTheorem[{0, 0, 1}, {4, 81, 25}]
```

```
2025
6400
7776
```

Итак, $u \equiv 2025 \pmod{8100}$, $v \equiv 6400 \pmod{8100}$, $w \equiv 7776 \pmod{8100}$. *Этим завершена предварительная работа.*

Решение уравнения (8.1) для $c = 7531, q = 8101$.

Сначала, пользуясь объясненным выше методом, вычислим $m^{(i)} = m \bmod p_i^{n_i}$, $1 \le i \le 3$, *как они определены в* (8.3). *Разумеется, на каждом шаге мы должны обращаться к ранее созданным таблицам.*

Первый простой делитель: $p_1 = 2, n_1 = 2$.

$$\begin{aligned} c &= & &= 7531, & c^{(8101-1)/2} &= 8100, & m_0 &= 1, \\ c_1 &= c \cdot \alpha^{-1} & &= 8006, & c_1^{(8101-1)/2^2} &= 1, & m_1 &= 0. \end{aligned}$$

Следовательно, $m^{(1)} = 1 + 0 \cdot 2^1 = 1$.

Второй простой делитель: $p_2 = 3, n_2 = 4$.

$$\begin{aligned} c &= & &= 7531, & c^{(8101-1)/3} &= 2217, & m_0 &= 2, \\ c_1 &= c \cdot \alpha^{-2} & &= 6735, & c_1^{(8101-1)/3^2} &= 1, & m_1 &= 0, \\ c_2 &= c_1 & &= 6735, & c_2^{(8101-1)/3^3} &= 2217, & m_2 &= 2, \\ c_3 &= c_2 \cdot \alpha^{-2 \cdot 3^2} & &= 6992, & c_3^{(8101-1)/3^4} &= 5883, & m_3 &= 1. \end{aligned}$$

Следовательно, $m^{(2)} = 2 + 0 \cdot 3^1 + 2 \cdot 3^2 + 1 \cdot 3^3 = 47$.

Третий простой делитель: $p_3 = 5, n_3 = 2$.

$$\begin{aligned} c &= & &= 7531, & c^{(8101-1)/5} &= 5221, & m_0 &= 4, \\ c_1 &= c \cdot \alpha^{-4} & &= 7613, & c_1^{(8101-1)/5^2} &= 356, & m_1 &= 2. \end{aligned}$$

Следовательно, $m^{(3)} = 4 + 2 \cdot 2^1 = 14$.

Окончательным решением m *будет* $m \equiv u \cdot m^{(1)} + v \cdot m^{(2)} + w \cdot m^{(3)} \equiv 6689 \pmod{8100}$. *В самом деле,*

```
Mod[2025 * 1 + 6400 * 47 + 7776 * 14, 8100]
```

|| 6689

Это легко проверяется:

```
PowerMod[6, 6689, 8101]
```

|| 7531

В "Mathematica" предвычисления u, v и w не обязательны, так как m можно прямо вычислить из $m^{(1)}$, $m^{(2)}$ и $m^{(3)}$ с помощью функции ChineseRemainderTheorem:

```
ChineseRemainderTheorem[{1, 47, 14}, {4, 81, 25}]
```

|| 6689

Если $q-1$ имеет большие простые делители, то доминирующими составляющими в объеме работы алгоритма Похлига–Хеллмана будут число $\sum_{i=1}^{k} p_i$ возведений в степень, необходимых для генерации таблиц $\{1, \alpha_i, \ldots, \alpha_i^{p_i-1}\}$, $1 \le i \le k$, и число $\sum_{i=1}^{k} n_i$ возведений в степень, необходимых для определения всех $m^{(i)}$.

В следующем подразделе мы обсудим метод взятия логарифмов, когда хотя бы одна степень простого числа, делящая $q-1$, слишком велика для того, чтобы поместиться в таблицы метода Похлига–Хеллмана.

8.3.2 Метод "детский шаг—гигантский шаг"

Если одна (или более) из степеней простых чисел, делящих $q-1$, слишком велика для метода Похлига–Хеллмана, можно использовать метод[5], излагаемый ниже. Он дает пользователю полную свободу уравновешивать длину таблицы, которую мы хотим записать, и остающийся объем работы. Начнем с примера.

Пример 8.8

Рассмотрим уравнение $29^m \equiv 30 \pmod{97}$ *и допустим, что мы можем записать таблицу только с* 10 *элементами поля. Создадим таблицу из чисел* $29^i \bmod 97$ *для* $i = 0, 1, \ldots, 9$ *и вычислим* $29^{-1} \bmod 97$ *с помощью функций "Mathematica" Table, PowerMod, Gridbox и Transpose.*

```
q = 97; a = 29;
powers = Table[{PowerMod[29, i, q], i}, {i, 0, 9}];
Gridbox[Transpose[powers],
    RowLines -> True, ColumnLines -> True] // DisplayForm
x = PowerMod[a, -1, q]
```

[5] Его английское название: "baby-step giant-step method".— *Прим. ред.*

1	29	65	42	54	14	18	37	6	77
0	1	2	3	4	5	6	7	8	9

87

Мы также нашли, что $29^{-1} \equiv 87 \pmod{97}$.

Записывая $m = 10j + i, 0 \leq i \leq 9$, *мы видим, что сравнение* $29^m \equiv 30 \pmod{97}$ *может быть переписано как* $29^i \equiv 30 \cdot 29^{-10j} \pmod{97}$ *или как* $29^i \equiv 30 \cdot 87^{10j} \pmod{97}$. *Поскольку* $87^{10} \equiv 49 \pmod{97}$, *получаем эквивалентную задачу решения сравнения* $29^i \equiv 30 \cdot 49^j \pmod{97}$, $0 \leq \leq i \leq 9$.

Сделаем это, испытывая $j = 0, 1, \ldots$ *и проверяя каждый раз, встречается ли* $30 \cdot 49^j \bmod 97$ *в списке степеней* $\{1, 29, 29^2, \ldots, 29^9\} \bmod 97$.

Чтобы облегчить просмотр таблицы степеней, отсортируем ее элементы посредством функции Sort.

```
sortedpowers = Sort[powers];
Gridbox[Transpose[sortedpowers],
    RowLines -> True, ColumnLines -> True] // DisplayForm
```

1	6	14	18	29	37	42	54	65	77
0	8	5	6	1	7	3	4	2	9

Далее мы испытываем $30 \cdot 49^j \bmod 97$, *пока результат не появится в нашей таблице. При этом используются функции* While, MemberQ *и* Mod *из пакета "Mathematica". Печатается также соответствующий столбец из таблицы сортированных степеней (и нужно уменьшить* j *на* 1, *потому что мы начинаем нумерацию* j *с* 0*).*

```
j = 0;
While[MemberQ[sortedpowers,
      {Mod[30 * 49^j, _}] == False, j = j + 1];
j
sortedpowers[[j - 1]]
```

4

{14, 5}

Мы получили, что $j = 4$ *и что* $30 \cdot 49^j \bmod 97$ *появляется в таблице как* 14, *т.е.* $29^5 \bmod 97$ *(и, значит,* $i = 5$*). Действительно,*

```
Mod[30 * 49^4, 97] == Mod[29^5, 97]
```

True

Это влечет, что $m = 10j + i = 10 \cdot 4 + 5 = 45$. *В самом деле,* $29^{45} \equiv 30 \pmod{97}$*; это можно легко проверить:*

```
PowerMod[29, 45, 97]
```

|| 30

Теперь сформулируем продемонстрированный метод в общем виде.

Теорема 8.1
Пусть α — примитивный элемент в $\mathrm{GF}(q)$, а p — делитель числа $q-1$ (не обязательно простой). Определим $\omega = \alpha^{(q-1)/p}$; тогда ω—примитивный корень степени p из единицы. Пусть c — произвольный корень степени p из единицы. Тогда для любого (компромиссного значения) t, $0 \le t \le 1$, можно найти показатель m, $0 \le m \le p-1$, удовлетворяющий равенству

$$c = \omega^m,$$

используя алгоритм, требующий

$p^{1-t} \cdot (1 + \log_2 p^t)$ операций,
$p^t \cdot \log_2 q$ битов памяти

и начальное вычисление, включающее

$p^t \cdot (1 + \log_2 p^t)$ операций.

Доказательство. Пусть $u = \lceil p^t \rceil$. Построим таблицу последовательных степеней ω^i, $0 \le i \le u-1$. Это требует $u \approx p^t$ умножений. Теперь отсортируем эту таблицу за $p^t \log_2 p^t$ операций (см. [Knut73], разд. 5.3). Вместе это дает число операций в предвычислении.

Каждый из $u \approx p^t$ элементов поля в таблице нуждается в $\log_2 q$ битах памяти. Это объясняет требования к памяти в теореме.

Введем i и j равенствами

$$m = j \cdot u + i, \qquad 0 \le i < u \approx p^t.$$

Заметим, что

$$0 \le j \le \frac{m}{u} < \frac{p}{u} \approx p^{1-t}.$$

Конечно, решение уравнения $c = \omega^m$ эквивалентно нахождению i и j, $0 \le \le i < u$, удовлетворяющих равенству

$$\omega^i = c \cdot \omega^{-j \cdot u}.$$

Чтобы решить последнее уравнение, мы просто вычисляем $c \cdot \omega^{-l \cdot u}$ для $l = 0, 1, \ldots$ и проверяем, появится ли результат в таблице. Это случится, когда $l = j$, значит, до того, как будет $l = \lceil p^{1-t} \rceil$. Для каждого значения l нужно выполнить одно умножение и поиск в таблице, который требует $\log_2 p^t$ операций.

■

При $t = 1/2$ данный алгоритм сводится к алгоритму, упомянутому в конце подразд. 8.1.1, со сложностью $\sqrt{q}$ (и по времени, и по памяти). А вот два крайних случая алгоритма:

$t = 0$: вообще нет таблиц; нужно испытывать все степени $1, \omega, \omega^2, \ldots$;

$t = 1$: создается полная таблица $1, \omega, \omega^2, \ldots, \omega^{q-1}$; нужен единственный просмотр таблицы.

Заметим, что в рассмотренном алгоритме произведение времени вычисления и числа битов памяти более или менее постоянно.

8.3.3 ρ-алгоритм Полларда

Временна́я сложность ρ-метода Полларда[6] [Poll78] такая же, как у метода "детский шаг–гигантский шаг", изложенного в предыдущем разделе. Преимущество же состоит в минимальных требованиях к памяти.

Мы объясним ρ-метод Полларда в специальном случае мультипликативной подгруппы G простого порядка из $\mathrm{GF}(q)$. Таким образом, мы хотим найти m, $0 \le m < p$, из уравнения $c = \alpha^m$ (см. (8.1)), где $\alpha \in \mathrm{GF}(q)$ имеет простой порядок p, а $c \in \mathrm{GF}(q)$ — заданный корень степени p из единицы. Заметим, что p делит $q-1$ по теореме B.5.

Пример 8.9 (часть 1)

Чтобы избежать вычислений в произвольном конечном поле, выберем в качестве q простое число 4679. *Отметим, что* $q - 1 = 2 \times 2339$. *Ниже мы убедимся, что* 11 *— примитивный элемент в* GF(4679) *и поэтому* $\alpha = 11^{(q-1)/2339} = 11^2 = 121$ *— образующий мультипликативной подгруппы порядка* 2339. *Все эти вычисления легко проверить с помощью функций PrimeQ, FactorInteger, PowerMod и функции MultiplicativeOrder, введенной в подразд.* B.1.1*:*

```
MultiplicativeOrder[a_, n_] := If[GCD[a, n] == 1,
        Divisors[EulerPhi[n]]//.
{x_, y_} -> If[PowerMod[a, x, n] == 1, x, {y}];
```

```
q = 4679;
PrimeQ[q]
FactorInteger[q - 1]
MultiplicativeOrder[11, q]
PowerMod[11, 2, q]
MultiplicativeOrder[121, q]
```

```
|| True
|| {{2, 1}, {2339, 1}}
|| 4678
|| 121
|| 2339
```

Ниже мы продолжим этот пример, когда будем решать уравнение

$$121^m \equiv 3435 \pmod{4679}.$$

Заметим, что это уравнение должно иметь решение, поскольку 3435 *действительно является корнем степени* 2339 *из единицы в* GF(4679).

[6] Английское название: "Pollard-ρ method".— *Прим. ред.*

В самом деле, все корни степени 2339 *из единицы суть нули многочлена* $x^{2339}-1$, *и других нулей по теореме* $B.15$ *у него нет.*

```
PowerMod[3435, 2339, 4679]
```

```
1
```

Чтобы решить уравнение $c=\alpha^m$, мы разбиваем мультипликативную подгруппу G порядка p из $\mathrm{GF}(q)$ на три подмножества G_i, $i=0,1,2$, следующим образом:

$$x\in G_i \iff x\equiv i \pmod 3.$$

Определим рекурсивно последовательность $\{x_i\}_{i\ge 0}$ в $\mathrm{GF}(q)$, положив $x_0=1$ и

$$x_{i+1}=f(x_i)=\begin{cases} x_i^2 \bmod q, & \text{если } x_i\in G_0,\\ (c\cdot x_i) \bmod q, & \text{если } x_i\in G_1,\\ (\alpha\cdot x_i) \bmod q, & \text{если } x_i\in G_2,\end{cases} \tag{8.4}$$

Сопоставим последовательности $\{x_i\}_{i\ge 0}$ две другие последовательности $\{a_i\}_{i\ge 0}$ и $\{b_i\}_{i\ge 0}$ так, чтобы для всех $i\ge 0$ выполнялось равенство

$$x_i=\alpha^{a_i}c^{b_i}.$$

С этой целью возьмем $a_0=b_0=0$ и вновь используем рекурсию:

$$a_{i+1}=\begin{cases} (2a_i) \bmod p, & \text{если } x_i\in G_0,\\ a_i, & \text{если } x_i\in G_1,\\ (a_i+1) \bmod p, & \text{если } x_i\in G_2.\end{cases}$$

$$b_{i+1}=\begin{cases} (2b_i) \bmod p, & \text{если } x_i\in G_0,\\ (b_i+1) \bmod p, & \text{если } x_i\in G_1,\\ b_i, & \text{если } x_i\in G_2.\end{cases}$$

Заметим, что по индукции

$$\begin{aligned} x_{i+1}&=x_i^2=(\alpha^{a_i}c^{b_i})^2=\alpha^{2a_i}c^{2b_i}=\alpha^{a_{i+1}}c^{b_{i+1}}, && \text{если } x_i\in G_0,\\ x_{i+1}&=c\cdot x_i=c\cdot\alpha^{a_i}c^{b_i}=\alpha^{a_i}c^{b_i+1}=\alpha^{a_{i+1}}c^{b_{i+1}}, && \text{если } x_i\in G_1,\\ x_{i+1}&=\alpha\cdot x_i=\alpha\cdot\alpha^{a_i}c^{b_i}=\alpha^{a_i+1}c^{b_i}=\alpha^{a_{i+1}}c^{b_{i+1}}, && \text{если } x_i\in G_2.\end{aligned}$$

Как только мы получаем два различных индекса i и j, для которых $x_i=x_j$, мы заканчиваем, поскольку это влечет $\alpha^{a_i}c^{b_i}=\alpha^{a_j}c^{b_j}$ и потому $\alpha^{a_i-a_j}=c^{b_j-b_i}$. При условии $b_i\ne b_j$ мы находим решение: $m\equiv(a_i-a_j)/(b_j-b_i) \pmod p$. Если же $b_i=b_j$, что случается с ничтожной вероятностью, положим $c'=c\cdot\alpha$ и решим уравнение $c'=\alpha^{m'}$, где $m'=m+1$.

Чтобы найти индексы i и j, для которых $x_i = x_j$, воспользуемся *алгоритмом Флойда нахождения цикла*: ищем такой индекс i, что $x_i = x_{2i}$ (т.е. берем $j = 2i$).

С этой целью мы начинаем с пары (x_1, x_2), находим (x_2, x_4), затем (x_3, x_6) и т.д., каждый раз вычисляя (x_{i+1}, x_{2i+2}), исходя из ранее вычисленной пары (x_i, x_{2i}), по следующим правилам: $x_{i+1} = f(x_i)$ и $x_{2i+2} = f^2(x_{2i})$. Таким путем можно избежать огромных записей.

Пример 8.9 (часть 2)

Продолжаем пример 8.9. *Следовательно,* $q = 4679$, $\alpha = 121$ — *элемент простого порядка* $p = 2339$ *и* $c = 3435$. *Таким образом, мы имеем уравнение*

$$121^m \equiv 3435 \pmod{4679}.$$

Рекуррентное соотношение для последовательности $\{x_i\}_{i\geq 0}$ *можно формализовать посредством функций* Which *и* Mod*:*

```
RecX[x_, alp_, c_, q_] := Which[Mod[x, 3] == 0, Mod[x^2, q],
    Mod[x, 3] == 1, Mod[c * x, q], Mod[x, 3] == 2,
    Mod[alp * x, q]]
```

Наименьший индекс i, *для которого* $x_i = x_{2i}$, *можно легко найти с помощью функции* While*:*

```
alp = 121; c = 3435; q = 4679;
x1 = RecX[1, alp, c, q];
x2 = RecX[x1, alp, c, q]; i = 1;
While[x1 != x2, x1 = RecX[x1, alp, c, q];
    x2 = RecX[RecX[x2, alp, c, q], alp, c, q]; i = i + 1];
i
```

```
76
```

Итак, $x_{76} = x_{152}$ *и* $m \equiv (a_{152} - a_{76})/(b_{76} - b_{152}) \pmod{2339}$. *Однако выше мы не обновили значения* a_i *и* b_i. *Сделаем это сейчас.*

```
RecurrDef[{x_, a_, b_}] := Which[
    Mod[x, 3] == 0, {Mod[x^2, q], Mod[2 a, p], Mod[2 b, p]},
    Mod[x, 3] == 1, {Mod[c * x, q], a, Mod[b + 1, p]},
    Mod[x, 3] == 2, {Mod[alp * x, q], Mod[a + 1, p], b}

alp = 121; c = 3435; q = 4679; p = 2339;
x1 = 1; a1 = 0; b1 = 0;
x2 = 1; a2 = 0; b2 = 0;
{x1, a1, b1} = RecurrDef[{x1, a1, b1}]; i = 1;
{x2, a2, b2} = RecurrDef[RecurrDef[{x2, a2, b2}]];
```

```
While[x1 != x2, {x1, a1, b1} = RecurrDef[{x1, a1, b1}];
    {x2, a2, b2} = RecurrDef[RecurrDef[{x2, a2, b2}]];
    i = i + 1];
Print["i=", i]
Print["\!\(x\_i\)=", x1, ", \!\(a\_i\)=", a1,
    ", \!\(b\_i\)=", b1]
Print["\!\(x\_\(2. i\)\)=", x2, ", \!\(a\_\(2. i\)\)=", a2,
    ", \!\(b\_\(2. i\)\)=", b2]
```

```
i=76
x_i=492, a_i=84, b_i=2191
x_2i=492, a_2i=286, b_2i=915
```

В самом деле, выражение $\alpha^{a_i} c^{b_i}$ *принимает одно и то же значение при* $i = 76$ *и* $i = 152$.

```
Mod[PowerMod[alp, a1, q] * PowerMod[c, b1, q], q]
Mod[PowerMod[alp, a2, q] * PowerMod[c, b2, q], q]
```

```
492
```

```
492
```

Теперь решение m *уравнения* $121^m \equiv 3435 \pmod{4679}$ *можно определить из сравнения* $m \equiv (286 - 84)/(2191 - 915) \pmod{2339}$*:*

```
m = Mod[(a2 - a1) * PowerMod[b1 - b2, -1, p], p]
```

```
1111
```

То, что $m = 1111$ *— действительно решение, можно проверить:*

```
PowerMod[alp, 1111, q] == c
```

```
True
```

Символ ρ в названии алгоритма отражает форму последовательности $\{x_i\}_{i \geq 0}$: "хвост" — начало последовательности до вхождения в цикл. Требования к памяти у алгоритма Флойда нахождения цикла действительно минимальны. Ожидаемое время его пробега — $\sqrt{p}$. За дальнейшими деталями читатель отсылается к работе [Poll78].

8.3.4 Метод исчисления индексов

□ Общее обсуждение

Чтобы описать метод исчисления индексов[7], рассмотрим циклическую группу G порядка N, порожденную элементом g. Таким образом, $G = \{e, g, g^2, \ldots, g^{N-1}\}$ и $g^N = e$. В этих условиях мы хотим найти m из уравнения $g^m = h$ (см. (8.1)) для заданного $h \in G$.

В основе метода исчисления индексов лежат следующие шаги.

[7] Английское название: "index-calculus method". Заметим, что в теории конечных полей "индекс" — синоним "дискретного логарифма".— *Прим. ред.*

1) Выбираем подходящее подмножество S из G с тем свойством, что большая доля элементов из G может быть выражена эффективным образом в виде произведения элементов из S. Множество S называется *базой делителей*. Элемент $a \in G$, который может быть представлен как произведение элементов из S, называется *гладким* относительно S. В следующих двух шагах элементы из S будут записаны в виде степеней g.

2) Находим достаточно большую совокупность I показателей i, таких, что каждая степень $g^i, i \in I$, может быть эффективно представлена в виде произведения элементов из S: $g^i = s_1^{u_{i,1}} s_2^{u_{i,2}} \dots s_k^{u_{i,k}}$. Взяв $\log_g$ от обеих частей, получим множество сравнений

$$i \equiv u_{i,1} \log_g s_1 + u_{i,2} \log_g s_2 + \dots + u_{i,k} \log_g s_k \pmod{N}, i \in I.$$

3) Рассматривая числа $\log_g s_j$, $1 \le j \le k$, как неизвестные, решаем указанную выше систему линейных сравнений (для этого система линейных сравнений должна иметь ранг k и множество I должно быть достаточно большим).

4) Выбираем случайный показатель r и пытаемся выразить $g^r h$ в виде произведения элементов из S. Как только это происходит, скажем, $g^r h = s_1^{\nu_1} s_2^{\nu_2} \dots s_k^{\nu_k}$, мы опять берем log_g от обеих частей и получаем

$$r + m \equiv \nu_1 \log_g s_1 + \nu_2 \log_g s_2 + \dots + \nu_k \log_g s_k \pmod{N}.$$

Так как значения всех $\log_g s_i$ уже определены на шаге 3 и r выбрано, из полученного сравнения определяем m.

Заметим, что целью шагов 2 и 3 служит решение проблемы логарифмов для элементов базы делителей. Шаг 4 пытается свести исходную проблему логарифма к таковой для элементов базы делителей.

Должно быть ясно, что размер базы делителей S является компромиссом между требованиями к возможностям памяти и вероятностью того, что случайный элемент из G (а именно $g^r h$) можно представить как произведение элементов из S.

В общем, при изложенном подходе имеются две (связанные друг с другом) нерешенные проблемы.

- Как определить хорошую базу делителей?
- Как выразить элемент из G в виде произведения элементов из S?

Ниже мы продемонстрируем изложенный метод в двух частных случаях, когда в ответ на сформулированные вопросы можно сказать больше, чем в общем случае.

Сложность

Существует много вариаций метода исчисления индексов. Как правило, их сложность растет субэкспоненциально относительно $\log_2 N$, тогда как методы, описанные в подразд. 8.3.1, 8.3.2 и 8.3.3, экспоненциальны по $\log_2 N$.

□ $\mathbb{Z}_p^*$, мультипликативная группа поля GF(p)

В данном случае $G = \{1, 2, \ldots, p-1\}$. Пусть g—образующий этой группы.

Выбор базы делителей S: первые k простых чисел $p_1, p_2, \ldots, p_k$.

Если k достаточно велико, большую долю элементов из G можно выразить в виде произведений степеней этих k простых чисел, т.е. такие элементы будут гладкими относительно S.

Техника выражения элемента из G в виде произведения элементов из S: делим данный элемент на все p_i.

Сложность

Адлеман в [Adle79] детально анализирует указанную технику и получает сложность

$$\exp\{C\sqrt{\ln p \ln\ln p}\}$$

для некоторой константы C.

Пример 8.10

Рассмотрим $\mathbb{Z}_{541}^*$ с примитивным элементом $g = 2$. Простоту числа 541 и примитивность элемента 2 можно проверить с помощью функций PrimeQ, FactorInteger и PowerMod пакета "Mathematica". В самом деле, по теореме B.5 порядок элемента 2 делит $|\mathbb{Z}_{541}^*| = 540$; поэтому нужно лишь проверить, что $2^{(p-1)/d} \not\equiv 1 \pmod{541}$ для любого простого делителя d числа $p - 1 = 540$.

```
p = 541;
PrimeQ[p]
FactorInteger[p - 1]
```

```
True
{{2, 2}, {3, 3}, {5, 1}}
```

```
PowerMod[2, (541 - 1) / 2, p]
PowerMod[2, (541 - 1) / 3, p]
PowerMod[2, (541 - 1) / 5, p]
```

```
540
129
48
```

В качестве базы делителей S возьмем множество из первых пяти простых чисел; они могут быть порождены функциями Prime и Table из пакета "Mathematica".

```
Table[Prime[i], {i, 1, 5}]
```

```
|| {2, 3, 5, 7, 11}
```

Мы хотим записать каждый из элементов этой базы делителей в виде степени элемента $g = 2$, т.е. решить проблему логарифмов для элементов базы делителей. С этой целью мы пытаемся найти степени элемента $g = 2$ в $\mathbb{Z}_{541}^*$, которые могут быть представлены как произведения элементов из множества $\{2, 3, 5, 7, 11\}$. Для этого можно использовать функции FactorInteger и PowerMod из пакета "Mathematica". Сделав попытку,

```
p = 541;
try = PowerMod[2, 102, p];
FactorInteger[try]
```

```
|| 136
```

```
|| {{2, 3}, {17, 1}}
```

мы видим, что полного разложения по $\{2, 3, 5, 7, 11\}$ нет. После нескольких попыток (и ошибок) находим подходящие нам элементы 2^{14}, 2^{81}, 2^{207}, 2^{214} и 2^{300}.

```
p = 541;
FactorInteger[PowerMod[2, 14, p]]
FactorInteger[PowerMod[2, 81, p]]
FactorInteger[PowerMod[2, 207, p]]
FactorInteger[PowerMod[2, 214, p]]
FactorInteger[PowerMod[2, 300, p]]
```

```
|| {{2, 1}, {7, 1}, {11, 1}}
```

```
|| {{2, 1}, {3, 1}, {7, 2}}
```

```
|| {{5, 2}, {11, 1}}
```

```
|| {{5, 1}, {7, 1}}
```

```
|| {{2, 5}, {11, 1}}
```

Полагая $m_1 = \log_2 2$, $m_2 = \log_2 3$, $m_3 = \log_2 5$, $m_4 = \log_2 7$, $m_5 = \log_2 11$ и беря логарифмы, получаем 5 линейных сравнений относительно $m_1, m_2, \ldots, m_5$.

Например, сравнение $2^{207} \equiv 5^2 \cdot 11^1 \pmod{541}$ можно записать в виде

$$2^{207} \equiv 2^{2\cdot\log_2 5} 2^{1\cdot\log_2 11} \equiv 2^{2m_3} 2^{m_5} \pmod{541}.$$

Взяв $\log_2$ *от обеих его частей, получим*

$$207 \equiv 2m_3 + m_5 \pmod{540}.$$

Таким образом, мы имеем

$$\begin{aligned} 14 &\equiv m_1 + m_4 + m_5 \pmod{540}, \\ 81 &\equiv m_1 + m_2 + 2m_4 \pmod{540}, \\ 207 &\equiv 2m_3 + m_5 \pmod{540}, \\ 214 &\equiv m_3 + m_4 \pmod{540}, \\ 300 &\equiv 5m_1 + m_5 \pmod{540}. \end{aligned}$$

Эта система линейных сравнений может быть решена с помощью функции Solve*:*

```
m1 = .; m2 = .; m3 = .; m4 = .; m5 = .;
Solve[{m1 + m4 + m5 == 14, m1 + m2 + m4 == 81,
    2 * m3 + m5 == 207, m3 + m4 == 214, 5 * m1 + m5 == 300,
    Modulus == 540}, {m1, m2, m3, m4, m4}]
```

```
{{Modulus->540, m2->104, m3->496, m1->1, m4->258,
m5->295}}
```

Таким образом, мы знаем, что

$$m_1 = \log_2 2 = 1, \quad m_2 = \log_2 3 = 104, \quad m_3 = \log_2 5 = 496,$$
$$m_4 = \log_2 7 = 258, \quad m_5 = \log_2 11 = 295$$

или, эквивалентно,

$$2^1 \equiv 2 \pmod{541}, \quad 2^{104} \equiv 3 \pmod{541}, \quad 2^{496} \equiv 5 \pmod{541},$$
$$2^{258} \equiv 7 \pmod{541}, \quad 2^{295} \equiv 11 \pmod{541}.$$

Если полученные линейные сравнения линейно зависимы, то некоторые сравнения нужно заменить другими, чтобы получилась система линейно независимых сравнений.

Найдем теперь решение уравнения $2^m \equiv 345 \pmod{541}$. *Из*

```
FactorInteger[345]
FactorInteger[Mod[2^2 * 345, 541]]
FactorInteger[Mod[2^100 * 345, 541]]
FactorInteger[Mod[2^13 * 345, 541]]
```

```
{{3, 1}, {5, 1}, {23, 1}}
```

```
{{2, 1}, {149, 1}}
```

```
{{3, 2}, {41, 1}}
```

```
{{2, 3}, {7, 1}}
```

видно, что число 345 *нельзя представить в виде произведения элементов из* S, *как нельзя этого сделать для* $2^2 \cdot 345$ *и* $2^{100} \cdot 345$, *но можно для* $2^{13} \cdot 345 = 2^3 7^1$ *в* GF(541).

Мы получаем

$$13 + m \equiv 3 \cdot m_1 + 1 \cdot m_4 \equiv 3 \cdot 1 + 258 \equiv 261 \pmod{540},$$

значит, решение уравнения $2^m \equiv 345 \pmod{541}$ *имеет вид*

$$m \equiv 248 \pmod{540}.$$

Это легко проверить:

```
PowerMod[2, 248, 541]
```

```
345
```

Ввиду малых параметров можно явно найти все элементы из $\{1, 2, \ldots, 540\}$, *которые представимы как произведения элементов из* S. *Мы используем функции* Select, Flatten, Table, Sort *и* Length *из пакета "Mathematica", а также тот факт, что в разложении любого числа, меньшего, чем* 541, *показатель двойки не превосходит* $\lfloor \log_2 541 \rfloor = 9$, *показатель тройки не превосходит* $\lfloor \log_3 541 \rfloor = 5$ *и т.д.*

```
BaseProd = Select[
    Flatten[Table[2^i1 3^i2 5^i3 7^i4 11^i5,
        {i1, 0, Log[2, 541]},
        {i2, 0, Log[3, 541]},
        {i3, 0, Log[5, 541]},
        {i4, 0, Log[7, 541]},
        {i5, 0, Log[11, 541]}]],
      # < 541 &] // Sort
Length[BaseProd]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 18,
20, 21, 22, 24, 25, 27, 28, 30, 32, 33, 35, 36, 40, 42,
44, 45, 48, 49, 50, 54, 55, 56, 60, 63, 64, 66, 70, 72,
75, 77, 80, 81, 84, 88, 90, 96, 98, 99, 100, 105, 108,
110, 112, 120, 121, 125, 126, 128, 132, 135, 140, 144,
147, 150, 154, 160, 162, 165, 168, 175, 176, 180, 189,
192, 196, 198, 200, 210, 216, 220, 224, 225, 231, 240,
242, 243, 245, 250, 252, 256, 264, 270, 275, 280, 288,
294, 297, 300, 308, 315, 320, 324, 330, 336, 343, 350,
352, 360, 363, 375, 378, 384, 385, 392, 396, 400, 405,
420, 432, 440, 441, 448, 450, 462, 480, 484, 486, 490,
495, 500, 504, 512, 525, 528, 539, 540}
```

```
142
```

Таким образом, более четверти всех элементов из G представимы как произведения элементов из S. Это означает, что в среднем нужно сделать четыре попытки выбора r до тех пор, когда $g^r h$ можно будет выразить в виде произведения элементов из множества $\{2, 3, 5, 7, 11\}$.

□ $\mathbf{GF(2^n)}$

Все элементы в $\mathrm{GF}(2^n)$ могут быть представлены посредством бинарных многочленов степени меньше n от x по модулю неприводимого многочлена $f(x)$ (см. теорему B.16). Поэтому $\mathrm{GF}(2^n) = \mathrm{GF}(2)[x]/(f(x))$.

Пусть многочлен $\alpha = \alpha(x)$ обозначает примитивный элемент в $\mathrm{GF}(2^n)$. Тогда $\mathrm{GF}(2^n)$ можно также представить бинарными многочленами степени меньше n по модулю минимального многочлена $p(x)$ элемента α. Это влечет, что α — примитивный элемент в $\mathrm{GF}(2)[\alpha]/(p(\alpha))$ и, значит, x — примитивный элемент в $\mathrm{GF}(2)[x]/(p(x))$.

См. пример B.6, где $f(x) = x^4 + x^3 + x^2 + x + 1$ определяет $\mathrm{GF}(2^4)$ и где $\alpha(x) = 1 + x$ — примитивный элемент в $\mathrm{GF}(2)[x]/(x^4 + x^3 + x^2 + x + 1)$. Этот α является нулем примитивного многочлена $p(x) = x^4 + x^3 + 1$. В $\mathrm{GF}(2)[x]/(x^4 + x^3 + 1)$ элемент x примитивен.

Задачу решения уравнения (8.1) можно переформулировать так:

для любого многочлена $c(x)$ степени меньше n найти показатель m,
$0 \le m \le 2^n - 2$, такой, что $x^m \equiv c(x) \pmod{p(x)}$.

В качестве базы делителей S выберем все бинарные неприводимые многочлены степени не больше t, скажем, $p_1(x), p_2(x), \ldots, p_k(x)$. (Число таких многочленов дает теорема B.17.)

Что касается техники представления элемента из $\mathrm{GF}(2^n)$ как произведения элементов из S, то мы просто делим данный элемент на все многочлены $p_i(x)$. Многочлен $u(x)$, который можно выразить в виде произведения элементов из S, называется *гладким* относительно S.

Сложность

Копперсмит [Copp84] проанализировал этот алгоритм и нашел асимптотическое время его работы:

$$\exp\{C\sqrt[3]{(\ln n)(\ln\ln n)^2}\}.$$

Позже были найдены улучшения алгоритма, известные под названиями *number field sieve* (решето числового поля) и *function field sieve* (решето поля многочленов) (см. [AdDM93], [Adle94], [HelR83]). Мы также отсылаем читателя к блестящему обзору [Odly85] по проблеме дискретных логарифмов.

Пример 8.11

Мы хотим взять логарифм в $\mathrm{GF}(2^{10})$. *Чтобы представить* $\mathrm{GF}(2^{10})$ *подходящим образом и найти в нем примитивный элемент, мы ищем примитивный многочлен степени* 10, *используя функцию* FieldIrredu-

cible из пакета "Mathematica". Для этого сначала загружается пакет Algebra'FiniteFields'.

```
<< Algebra'FiniteFields'
```

```
fld = GF[2, 10];
FieldIrreducible[fld, x]
```

|| $1 + x^7 + x^{10}$

Итак, мы берем $\mathrm{GF}(2^{10}) = \mathrm{GF}(2)[x]/(x^{10} + x^7 + 1)$ *с примитивным элементом* x. *Теперь уравнение* (8.1) *читается следующим образом:*

$$\text{найти такое } m, \text{ что } x^m \equiv c(x) \pmod{x^{10} + x^7 + 1}.$$

В качестве базы делителей S *возьмем множество всех неприводимых многочленов степени не больше* 4. *Читатель может вспомнить, что все бинарные неприводимые многочлены степени* d *входят в разложение многочлена* $x^{2^d} - x$ *(см. теорему* B.35*).*

```
Clear[x];
Factor[x^(2^3) - x, Modulus -> 2]
Factor[x^(2^4) - x, Modulus -> 2]
```

|| $x\ (1 + x)\ (1 + x + x^3)\ (1 + x^2 + x^3)$

|| $x\ (1 + x)\ (1 + x + x^2)\ (1 + x + x^4)\ (1 + x + x^2 + x^3 + x^4)$

Следовательно, база делителей S *такова:*

$$\begin{aligned} p_1(x) &= x, & p_2(x) &= 1 + x, \\ p_3(x) &= 1 + x + x^2, & p_4(x) &= 1 + x + x^3, \\ p_5(x) &= 1 + x^2 + x^3, & p_6(x) &= 1 + x + x^2 + x^3 + x^4, \\ p_7(x) &= 1 + x + x^4, & p_8(x) &= 1 + x^3 + x^4. \end{aligned}$$

Мы хотим записать каждый из элементов базы делителей в виде степени x, *т.е. решить проблему логарифмов для элементов базы делителей. С этой целью попытаемся найти те степени* x *в* $\mathrm{GF}[x]/(x^{10} + x^7 + 1)$, *которые выражаются в виде произведений многочленов* $p_j(x)$, $1 \le j \le 8$. *При этом используются функции* Factor *и* PolynomialMod *из пакета "Mathematica".*

```
attempt = PolynomialMod[x^85, {x^10 + x^7 + 1, 2}];
Factor[attempt, Modulus -> 2]
```

|| $1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^9$

|| $(1 + x + x)^2\ (1 + x + x^2)\ (1 + x^2 + x^3 + x^4 + x^5)$

Заключаем, что степень x^{85} *не гладка относительно нашей базы делителей. После нескольких попыток (и ошибок) находим следующий список гладких степеней* x:

```
Factor[PolynomialMod[x, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^86, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^140, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^211, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^319, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^457, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^605, {x^10 + x^7 + 1, 2}], Modulus -> 2]
Factor[PolynomialMod[x^787, {x^10 + x^7 + 1, 2}], Modulus -> 2]
```

```
x
(1 + x + x^3) (1 + x^2 + x^3)
x^2 (1 + x + x^2)^2
(1 + x)^5 (1 + x + x^2 + x^3 + x^4)
(1 + x) (1 + x^3 + x^4) (1 + x + x^2 + x^3 + x^4)
(1 + x + x^2) (1 + x + x^3) (1 + x + x^4)
(1 + x) (1 + x^2 + x^3) (1 + x + x^4)
(1 + x + x^3) (1 + x^2 + x^3)^2
```

Записывая $p_i(x) \equiv x^{m_i} \pmod{x^{10}+x^7+1}$, *мы в итоге получим восемь линейных сравнений. Например, последний ответ выше дает*

$$x^{787} \equiv (1+x+x^3)(1+x^2+x^3)^2 \equiv x^{m_4}(x^{m_5})^2 \equiv x^{m_4+2m_5} \pmod{x^{10}+x^7+1}.$$

Взяв логарифмы обеих частей, приходим к линейному сравнению

$$787 \equiv m_4 + 2m_5 \pmod{1023},$$

так как 1023 — *мультипликативный порядок примитивного элемента* x. *На этом пути мы и получаем восемь линейных сравнений:*

$$\begin{aligned}
1 &\equiv m_1 \pmod{1023},\\
86 &\equiv m_4 + m_5 \pmod{1023},\\
140 &\equiv 2m_1 + 2m_3 \pmod{1023},\\
211 &\equiv 5m_2 + m_6 \pmod{1023},\\
319 &\equiv m_2 + m_6 + m_8 \pmod{1023},\\
457 &\equiv m_3 + m_4 + m_7 \pmod{1023},\\
605 &\equiv m_2 + m_5 + m_7 \pmod{1023},\\
787 &\equiv m_4 + 2m_5 \pmod{1023}.
\end{aligned}$$

Эту систему линейных сравнений можно решить с помощью функции Solve *пакета "Mathematica".*

```
Clear[m1, m2, m3, m4, m5, m6, m7, m8];
Solve[{m1 == 1, m4 + m5 == 86, 2 m1 + 2 m3 == 140,
  5 m2 + m6 == 211, m2 + m6 + m8 == 319, m3 + m4 + m7 == 457,
  m2 + m5 + m7 == 605, m4 + 2 m5 == 787, Modulus == 1023},
  {m1, m2, m3, m4, m5, m6, m7, m8}]
```

```
{{Modulus -> 1023, m8 -> 827, m1 -> 1, m3 -> 69,
m6 -> 591, m7 -> 1003, m2 -> 947, m4 -> 408, m5 -> 701}}
```

Тем самым мы узнали, что $m_1 = 1$, $m_2 = 947$, $m_3 = 69$, $m_4 = 408$, $m_5 = 701$, $m_6 = 591$, $m_7 = 1003$, $m_8 = 827$.

Если линейные сравнения оказываются линейно зависимыми, нужно некоторые из них заменить другими, добиваясь линейной независимости.

Найдем теперь решение уравнения $x^m \equiv 1 + x + x^6 + x^9 \pmod{x^{10} + x^7 + 1}$. *Из*

```
Factor[PolynomialMod[
    1 + x + x^6 + x^9, {x^10 + x^7 + 1}], Modulus -> 2]
Factor[PolynomialMod[
    x^50 (1 + x + x^6 + x^9), {x^10 + x^7 + 1}], Modulus -> 2]
```

```
(1 + x)^2 (1 + x + x^2 + x^3 + x^4 + x^5 + x^7)
(1 + x + x^2)^2 (1 + x + x^4)
```

видно, что $1 + x + x^6 + x^9$ *нельзя записать как произведение многочленов из* S, *но* $x^{50}(1 + x + x^6 + x^9)$ — *можно.*

Мы получили, что $50 + m \equiv 2m_3 + m_7 \equiv 2 \cdot 69 + 1003 \equiv 118 \pmod{1023}$, *так что решением уравнения* $x^m \equiv 1 + x + x^6 + x^9 \pmod{x^{10} + x^7 + 1}$ *служит*

$$m \equiv 68 \pmod{1023}.$$

Это легко проверить:

```
PolynomialMod[x^68, {x^10 + x^7 + 1, 2}]
```

```
1 + x + x^6 + x^9
```

8.4 Задачи

Задача 8.1^M. Пользователи A и B хотят применить систему Диффи–Хеллмана, чтобы установить общий ключ по публичному каналу. Они используют GF(p) с $p = 541$ и примитивным элементом $\alpha = 2$. Пользователь B делает публичным $c_B = 123$. Если $m_A = 432$, то каков общий ключ $k_{A,B}$, который A и B используют для связи?

Задача 8.2. Пользователи A и B хотят применить систему Диффи– Хеллмана, чтобы установить общий ключ по публичному каналу. Они используют

$\mathbb{F}_2[x]/(x^{10}+x^3+1)$ для представления $\mathrm{GF}(2^{10})$. Пользователь B делает публичным $c_B = 0100010100$, обозначающее элемент поля $x + x^5 + x^7$. Если $m_A = 2$, то каков общий ключ, который A и B используют для связи?

Задача 8.2. Продемонстрируйте версию алгоритма Похлига–Хеллмана, которая вычисляет логарифмы в конечном поле размера $q = 2^n + 1$, взяв $\log_2 142$ в $\mathrm{GF}(257)$.

Задача 8.4^M. Проверьте, что 953 — простое число и что 3 — образующий в $\mathbb{Z}^*_{953}$. Найдите три наименее значащих бита решения m сравнения $3^m \equiv 726 \pmod{953}$. (См. замечание в обсуждении частного случая $q - 1 = 2^n$.)

Задача 8.5^M. Вычислите $\log_3 135$ в $\mathrm{GF}(353)$ с помощью алгоритма Похлига–Хеллмана.

Задача 8.6^M. Найдите $\log_{44} 55$ в $\mathrm{GF}(197)$ методом "детский шаг—гигантский шаг", когда в таблицу могут быть записаны только 15 элементов поля.

Задача 8.7^M. Проверьте, что $\alpha = 662$ — примитивный корень степени 2003 из единицы в $\mathrm{GF}(4007)$ (заметим, что число 4007 просто). Пусть G — мультипликативная подгруппа порядка 2003 в $\mathrm{GF}(4007)$, порожденная α. Проверьте, что 2124 лежит в G.

Задача 8.8^M. Проверьте, что $g = 996$ — образующий мультипликативной группы $\mathbb{Z}^*_{4007}$. Применив метод исчисления индексов с базой делителей размера 6, определите $\log_{996} 1111$.

Задача 8.9^M. Решите уравнение $x^m \equiv 1 + x^3 + x^9 \pmod{x^{10} + x^3 + 1}$ по образцу примера 8.11.

Задача 8.10^M. Какова вероятность того, что случайный элемент $x^m \bmod (x^{10}+x^3+1)$ — гладкий относительно множества всех бинарных неприводимых многоленов степени не больше 10 (см. пример 8.1)?

Глава 9

Системы, основанные на методе RSA

9.1 Система RSA

В 1978 г. Райвест, Шамир и Адлеман [RivSA78] предложили криптосистему с публичными ключами, которая стала известна, как система RSA. Она использует следующие три факта:

1) возведение в степень по модулю составного числа n, т.е. вычисление c из сравнения $c \equiv m^e \pmod{n}$ для заданных m и e — относительно простая операция (см. подразд. 8.1.1);

2) противоположная задача извлечения корней по модулю большого составного числа n, т.е. вычисление m из сравнения $c \equiv m^e \pmod{n}$ (которое можно записать в виде $m \equiv \sqrt[e]{c} \pmod{n}$) для заданных c и e, вообще говоря, чрезвычайно трудна;

3) если примарное разложение n известно, то задача извлечения корней по модулю n выполнима.

9.1.1 Немного математики

Процитируем теорему А.14 и определение А.6 функции Эйлера из приложения А.

Теорема 9.1 (*Эйлер*)
Пусть a и n — натуральные числа. Тогда

$$\text{НОД}(a, n) = 1 \Longrightarrow a^{\varphi(n)} \equiv 1 \pmod{n}, \tag{9.1}$$

где функция Эйлера $\varphi(n)$ подсчитывает количество натуральных чисел от 1 до n, взаимно простых с n. Функцию φ можно вычислить из равенства

$$\varphi(n) = n \prod_{p|n, p \text{ простое}} (1 - \frac{1}{p}). \tag{9.2}$$

Читатель может проверить это с помощью функций *GCD* и *EulerPhi* пакета “Mathematica”.

```
n = 1999; a = 1234;
GCD[a, n]
ph = EulerPhi[n]
PowerMod[a, ph, n]
```

```
1
1998
1
```

9.1.2 Формирование системы

□ **Шаг 1: вычисление модуля n_U**

Каждый пользователь U системы выбирает два различных простых числа, скажем, p_U и q_U. В первоначальном предложении была рекомендована длина этих чисел порядка 100 (десятичных) цифр.

Пусть $n_U = p_U q_U$. Из равенства (9.2) следует, что

$$\varphi(n_U) = n_U(1 - \frac{1}{p_U})(1 - \frac{1}{q_U}) = (p_U - 1)(q_U - 1). \qquad (9.3)$$

Это можно увидеть и непосредственно. Из n_U целых чисел от 1 до $n_U = p_U q_U$ все взаимно просты с n_U, кроме q_U чисел, кратных p_U (именно $p_U, 2p_U, 3p_U, \ldots, q_U p_U$) и p_U чисел, кратных q_U (именно $q_U, 2q_U, 3q_U, \ldots, p_U q_U$). При подсчете нужно лишь осознать, что число $p_U q_U$ отбрасывалось слишком часто.

Пример 9.1 (часть 1)

Чтобы пример был "управляемым", участник Боб выбирает простые числа разумно малыми. Он использует функции Prime и EulerPhi из пакета "Mathematica".

```
pB = Prime[1200]
qB = Prime[1250]
nB = pB * qB
phiB = EulerPhi[nB]
```

```
9733
10177
99052741
99032832
```

□ **Шаг 2: Вычисление показателей e_U и d_U**

Пользователь U выбирает такое натуральное число e_U, $1 < e_U < \varphi(n_U)$, что НОД$(e_U, \varphi(n_U)) = 1$. Затем он вычисляет единственное d_U, удовлетворяющее условиям

$$e_U d_U \equiv 1 \pmod{\varphi(n_U)}, \quad 1 < d_U < \varphi(n_U). \qquad (9.4)$$

К примеру, U может использовать расширенный алгоритм Эвклида (см. разд. А.2), чтобы найти d_U менее чем за $\log_f \varphi(n_U)$ операций, где $f = (1+\sqrt{5})/2$ (теорема А.9).

Пример 9.1 (часть 2)

Случайный выбор числа e_B и вычисление d_B можно выполнить с помощью функций Random, While и ExtendedGCD пакета "Mathematica".

```
eB = Random[Integer, {1, nB}];
While[GCD[eB, phiB] !=1,
        eB = Random[Integer, {1, nB}]];
eB
ExtendedGCD[eB, phiB]
```

```
|| 81119923
|| {1, {17089915, -13998717}}
```

Таким образом, Боб имеет $e_B = 81119923$ и $d_B = 17089915$. Это можно проверить, вычисляя функцию Mod:

```
dB = 17089915;
Mod[eB * dB, phiB]
```

```
|| 1
```

□ **Шаг 3: публикация e_U и n_U**

Каждый пользователь U делает публичными числа e_U и n_U, сохраняя в секрете d_U. Простые числа p_U и q_U более не играют никакой роли. Но сам U может использовать их, как мы позже увидим, для уменьшения сложности своих вычислений. Поэтому p_U и q_U не могут быть опубликованы[1].

9.1.3 RSA как схема шифрования

Если пользователь A, скажем, Алиса, хочет послать секретное сообщение Бобу, она представляет свое сообщение в произвольной стандартизованной форме как число $m, 0 < m < n_B$. Затем Алиса отыскивает публичный показатель e_B Боба и посылает шифртекст c, вычисленный следующим образом:

$$c = m^{e_B} \bmod n_B.$$

Боб может восстановить m из c, возводя c в степень d_B, которая известна только ему. В самом деле, для некоторого целого l мы имеем

$$c^{d_B} \equiv (m^{e_B})^{d_B} \equiv m^{e_B d_B} \overset{(9.4)}{\equiv} m^{1+l\cdot\varphi(n_B)} \equiv m(m^{\varphi(n_B)})^l \overset{(9.1)}{\equiv} m \pmod{n_B}, \tag{9.5}$$

[1] Обычно в литературе пара (n_U, e_U) называется *публичным ключом*, а пара (n_U, d_U) (или одно число d_U) — *секретным ключом* пользователя U.— *Прим. ред.*

когда НОД$(m, n_B) = 1$. В задаче 9.2 читателю предлагается проверить, что система работает и тогда, когда НОД$(m, n_B) \neq 1$.

Резюмируем систему шифрования RSA в следующей таблице.

ПУБЛИЧНЫЕ	e_U И n_U ВСЕХ ПОЛЬЗОВАТЕЛЕЙ U
СЕКРЕТНОЕ	d_U ПОЛЬЗОВАТЕЛЯ U
СВОЙСТВО	$e_U d_U \equiv 1 \pmod{\varphi(n_U)}$
СООБЩЕНИЕ ДЛЯ B	$m, 0 < m < n_B$
ШИФРОВАНИЕ ДЛЯ B	$c = m^{e_B} \bmod n_B$
ДЕШИФРОВАНИЕ	$m = c^{d_B} \bmod n_B$

Таблица 9.1. RSA как схема шифрования.

Публичный и секретный показатели в системе RSA традиционно обозначаются через e_U и d_U, чтобы указать на функции шифрования (encryption) и дешифрования (decryption) соответственно.

Пример 9.1 (часть 3)

Мы продолжаем пример 9.1 с теми же параметрами, так что $n_B = 99052741$, $e_B = 81119923$ *и* $d_B = 17089915$. *Шифрование* $c = m^{e_B} \bmod n_B$ *сообщения* $m = 12345678$ *выполняется в пакете "Mathematica" с помощью функции* PowerMod*:*

```
nB = 99052741; eB = 81119923; dB = 17089915;
m = 12345678;
c = PowerMod[m, eB, nB]
```

```
38447790
```

Боб дешифрует это, вычисляя $c^{d_B} \bmod n_B$, *что дает* m.

```
PowerMod[c, dB, nB]
```

```
12345678
```

Можно уменьшить объем работы в процессе дешифрования, применив китайскую теорему об остатках (теорема А.19). В самом деле, Боб, зная разложение $n = p \times q$, может поступить следующим образом.

Он предвычисляет целые числа a и b, меньшие n, удовлетворяющие сравнениям

$$\begin{cases} a \equiv 1 \pmod{p}, \\ a \equiv 0 \pmod{q}; \end{cases}$$

$$\begin{cases} b \equiv 0 \pmod{p}, \\ b \equiv 1 \pmod{q}. \end{cases}$$

Получив шифртекст c, Боб вычисляет $m_1 \equiv c_1^d \bmod p$ и $m_2 \equiv$ $\equiv c_2^d \bmod q$, где $c_1 = c \bmod p$ и $c_2 = c \bmod q$. Заметим, что все эти вычисления выполняются по модулю p и q, типичная длина которых составляет половину длины n. По китайской теореме об остатках число $m = c^d \bmod n$ равно $(m_1 a + m_2 b) \bmod n$.

Этот подход приносит дополнительное преимущество. В силу теоремы Ферма (теорема А.15) показатель d при вычислении m_1 и m_2 может быть приведен по модулю $p-1$, соответственно, $q-1$. Действительно, $m_1 \equiv c^d \equiv c^{d_1} \pmod{p}$, где $d_1 = d \bmod (p-1)$, и аналогичное сравнение справедливо для вычислений по модулю q.

Пример 9.1 (часть 4)

Продолжаем с параметрами примера 9.1, так что $p_B = 9733$, $q_B = 10177$, $n_B = 99052741$, $e_B = 81119923$ *и* $d_B = 17089915$. *Чтобы найти решения систем*

$$\begin{cases} a \equiv 1 \pmod{9733}, \\ a \equiv 0 \pmod{10177}; \end{cases}$$

$$\begin{cases} b \equiv 0 \pmod{9733}, \\ b \equiv 1 \pmod{10177}. \end{cases}$$

загрузим пакет NumberTheory'NumberTheoryFunctions'.

```
<< NumberTheory`NumberTheoryFunctions`
```

и найдем числа a и b с помощью функции ChineseRemainderTheorem:

```
a = ChineseRemainderTheorem[{1, 0}, {9733, 10177}]
b = ChineseRemainderTheorem[{0, 1}, {9733, 10177}]
```

```
45287650
```

```
53765092
```

Далее, вычисляем $m_1 \equiv c^d \equiv c^{d_1} \pmod{p}$ *и* $m_2 \equiv c^d \equiv c^{d_2} \pmod{q}$, *получая*

```
p = 9733; q = 10177; d = 17089915;
c = 38447790;
c1 = Mod[c, p]
c2 = Mod[c, q]
d1 = Mod[d, p - 1]
d2 = Mod[d, q - 1]
m1 = Mod[c1, d1, p]
m2 = Mod[c2, d2, q]
```

```
2440
```

```
9261
```

```
523
```

```
|| 4411

|| 4234

|| 977
```

Результат дешифрования дается выражением $(m_1 a + m_2 b) \bmod n$ *и совпадает с указанным ранее.*

```
n = 99052741;
Mod[m1 * a + m2 * b, n]
```

```
|| 12345678
```

9.1.4 RSA как схема подписи

Равным образом систему RSA можно использовать для подписывания сообщений. Чтобы подписать сообщение m, $0 < m < n_B$, Боб вычисляет $c = m^{d_B} \bmod n_B$.

Получатель сообщения, скажем, Алиса, может легко восстановить исходное m, вычисляя $c^{e_B} \bmod n_B$, так как параметры e_B и n_B Боба публичны. Чтобы проверить это, повторим вычисления (9.5) (с небольшой вариацией):

$$c^{e_B} \equiv (m^{d_B})^{e_B} \equiv m^{e_B d_B} \overset{(9.4)}{\equiv} m^{1+l\cdot\varphi(n_B)} \equiv m(m^{\varphi(n_B)})^l \overset{(9.1)}{\equiv} m \pmod{n_B} \tag{9.6}$$

для всех m с НОД$(m, n_B) = 1$. Сравнение $c^{e_B} \equiv m \pmod{n_B}$ выполняется и тогда, когда НОД$(m, n_B) \neq 1$. В задаче 9.2 читателю предлагается доказать это.

Алиса должна рассматривать c как подпись Боба к m. Только Боб может вычислить c, исходя из m, потому что ему одному известно d_B. Мы советуем читателю перечитать обсуждение перед табл. 7.2.

ПУБЛИЧНЫЕ	e_U И n_U ВСЕХ ПОЛЬЗОВАТЕЛЕЙ U
СЕКРЕТНЫЙ	d_U ПОЛЬЗОВАТЕЛЯ U
СВОЙСТВО	$e_U d_U \equiv 1 \pmod{\varphi(n_U)}$
СООБЩЕНИЕ БОБА	$m, 0 < m < n_B$
B ПОДПИСЫВАЕТ	$c = m^{d_B} \bmod n_B$
A ПРОВЕРЯЕТ	$c^{e_B} \equiv m \pmod{n_B}$

Таблица 9.2. Система RSA для подписывания.

Пример 9.1 (часть 5)

Боб подписывает сообщение $m = 11111111$, *вычисляя* $c = m^{d_B} \bmod n_B$.

```
m = 11111111;
c = PowerMod[m, dB, nB]
```

```
74138899
```

Алиса проверяет подпись, вычисляя $c^{e_B} \bmod n_B$ *и получая* m.

9.1.5 RSA как схема шифрования и подписи

Предположим, что Алиса хочет подписать сообщение m Бобу. Общее решение, описанное в подразд. 7.1.4, а именно, когда Алиса сначала подписывает m с помощью своего секретного ключа, а затем шифрует результат с помощью публичного ключа Боба, не всегда непосредственно применимо в системе RSA.

Чтобы увидеть это, заметим, что Алиса должна была бы послать

$$c = (m^{d_A} \bmod n_A)^{e_B} \bmod n_B. \tag{9.7}$$

Однако это отображение не будет инъективным, если $n_A > n_B$. Например, оба сообщения $m_1 = 1$ и $m_2 = (1 + n_B)^{e_A}$ отобразятся в $c = 1$.

Так как Алиса и Боб не хотят делиться своими простыми числами, необходимо, чтобы было $n_A < n_B$. В этом случае Боб может восстановить m следующим образом:

$$(c^{d_B} \bmod n_B)^{e_A} \bmod n_A = m.$$

Чтобы проверить это, воспользуйтесь соотношениями (9.5) и (9.6).

Конечно, теперь возникает новая проблема — что делать, когда Боб захочет подписать конфиденциальное сообщение Алисе. Простое решение состоит в том, чтобы каждый пользователь создавал два набора параметров — один с модулем, меньшим некоторого порога T, и другой с модулем, большим T. В этих условиях отправитель использует свой собственный меньший модуль для подписи и больший модуль получателя — для шифрования.

ПУБЛИЧНЫЕ	e_{Ui} И n_{Ui} ВСЕХ ПОЛЬЗОВАТЕЛЕЙ U, $i = 1, 2$
СЕКРЕТНЫЕ	d_{Ui} ПОЛЬЗОВАТЕЛЯ U, $i = 1, 2$
СВОЙСТВА	$e_{Ui}d_{Ui} \equiv 1 \pmod{n_{Ui}}$, $n_{U1} < T < n_{U2}$
СООБЩЕНИЕ ОТ A К B	$m, 0 < m < n_{A1}$
A ПОСЫЛАЕТ	$c = (m^{d_{A1}} \bmod n_{A1})^{e_{B2}} \bmod n_{B2}$
B ВЫЧИСЛЯЕТ	$(c^{d_{B2}} \bmod n_{B2})^{e_{A1}} \bmod n_{A1} = m$
B СЧИТАЕТ ПОДПИСЬЮ	$c^{d_{B2}} \bmod n_{B2}$, РАВНОЕ $m^{d_{A1}} \bmod n_{A1}$

Таблица 9.3. RSA как схема шифрования и подписи.

Если между Алисой и Бобом возникает спор, то они идут к арбитру. Этот арбитр получает от Боба числа m и $u = c^{d_{B2}} \bmod n_{B2}$. Последнее равно $m^{d_{A1}} \bmod n_{A1}$, так как

$$c^{d_{B2}} \bmod n_{B2} \overset{(9.7)}{=} ((m^{d_{A1}} \bmod n_{A1})^{e_{B2}} \bmod n_{B2})^{d_{B2}} \overset{(9.5)}{=} m^{d_{A1}} \bmod n_{A1}.$$

Точно также, как в подразд. 9.1.4, арбитр проверяет, будет ли $u^{e_{A1}} \equiv m \pmod{n_{A1}}$. Если это условие выполняется, то сообщение m действительно пришло от Алисы; если же нет, u не будет рассматриваться как подпись Алисы к m.

Заметим, что арбитр не нуждается в знании секретных показателей Алисы или Боба. Поэтому Алиса и Боб могут продолжать использовать свои исходные наборы параметров.

9.2 Безопасность RSA: некоторые алгоритмы факторизации

9.2.1 Что может делать криптоаналитик

Предположим, что злоумышленник, скажем, Ева, перехватила секретное сообщение $c = m^{e_B} \bmod n_B$ для Боба. Если Ева знает секретный показатель d_B Боба, то она может вскрыть m из шифртекста c в точности тем же путем, что и Боб, а именно вычислив $c^{d_B} \bmod n_B$ (см. (9.5)).

Определить d_B по публичному показателю e_B и сравнению $e_B d_B \equiv 1 \pmod{\varphi(n_B)}$ (см. (9.4)) Еве легко, как только она знает $\varphi(n_B)$: она просто использует расширенный алгоритм Эвклида, как это делал Боб при формировании системы.

Чтобы найти $\varphi(n_B) = (p_B - 1)(q_B - 1)$ (см. (9.3)), Еве нужно факторизовать (т.е. разложить на простые множители) число n_B.

Во время введения RSA Шроппель (Schroeppel, не опубликовано) предложил модификацию алгоритма факторизации Моррисона и Бриллхарта [MorB75]. Она включала

$$e^{\sqrt{\ln n \ln\ln n}} \qquad \text{операций.}$$

При построении следующей таблицы, дающей представление о росте этого выражения, используются функции *TableForm*, *Table*, *Exp*, *Sqrt*, *Log* и *N* из пакета “Mathematica”.

```
TableForm[  Table[
  {k, N[Exp[Sqrt[Log[10^k]*Log[Log[10^k]]]], 3]},
  {k, 25, 250, 25}], TableHeadings ->
  {{}, {"length in digits", "complexity"}},
TableAlignments -> {Center}]
```

length in digits	complexity
25	4.3×10^6
50	1.42×10^{10}
75	8.99×10^{12}
100	2.34×10^{15}
125	3.41×10^{17}
150	3.26×10^{19}
175	2.25×10^{21}
200	1.2×10^{23}
225	5.17×10^{24}
250	1.86×10^{26}

Как можно видеть, если n имеет длину около 200 цифр, указанный выше криптоанализ недоступен. С другой стороны, были разложены числа, много бóльшие, чем считалось возможным в то время, когда была предложена схема RSA (при публикации этой книги рекордом были 512-битные числа). По этой причине очевидны предложения использовать для RSA существенно бóльшие модули.

Пример современного быстрого алгоритма факторизации можно найти в [LensH86]. Другие методы будут обсуждены в подразд. 9.2.3. Существуют специальные алгоритмы факторизации, которые работают быстрее, если n имеет особую форму. Один из таких методов мы изложим в следующем подразделе.

Кажется, до сих пор нет никакого способа взлома системы RSA, кроме разложения модуля n, но формальное доказательство эквивалентности этих двух проблем отсутствует. В разд. 9.5 мы обсудим вариант системы RSA, для которого можно показать, что его взлом эквивалентен разложению модуля.

Недостатком при выборе больших модулей является то, что выполнение одного возведения в степень требует больше времени, чем желательно, особенно когда нужно шифровать длинный файл. В такой ситуации часто применяют гибридную систему: для шифрования данных используется симметричная система с секретным ключом k, а схема RSA используется, чтобы безопасно послать этот ключ получателю (применив публичные параметры получателя).

При генерации p и q было бы плохой идеей сначала генерировать p, а затем испытывать на простоту числа $p+2, p+4, \ldots$. В действительности нужно, чтобы число $q-p$ (считая $p<q$) было большим. В самом деле, если криптоаналитик может угадать $q-p$, например, проверяя все вероятные значения, то он может определить $q+p$ из равенства

$$4n = 4pq = (q+p)^2 - (q-p)^2.$$

Из двух линейных уравнений находятся p и q, что влечет за собой взлом системы.

Пример 9.2

Пусть $n = 5007958289$. *Угадав, что* $q - p = 200$, *мы находим* $q + p$ *из*

```
n = 5007978289; Sqrt[4 * n + 200^2]
```

```
141534
```

Из уравнений $q+p = \sqrt{4n + 200^2}$ *и* $q-p = 200$ *выводим* $q = (\sqrt{4n + 200^2}+$ $+200)/2$.

```
q = (Sqrt[4 * n + 200^2] + 200) / 2
p = q - 200
```

```
70867
```

```
70667
```

```
p * q == n
```

```
True
```

Итак, число $|q - p|$ должно быть большим. Способ достичь этого — взять q больше, чем $p + \sqrt{p}$.

В литературе можно также найти несколько атак на систему RSA, имеющих вероятность успеха, которая лишь незначительно больше вероятности того, что случайно выбранное натуральное число, меньшее n, имеет нетривиальный делитель, общий с числом n. Тогда этот делитель — либо p, либо q. Вероятность того, что это произойдет, может быть вычислена с помощью функции Эйлера $\varphi(n)$:

$$\frac{n - \varphi(n)}{n} \overset{(9.3)}{=} \frac{p \cdot q - (p-1)(q-1)}{p \cdot q} = \frac{p + q - 1}{p \cdot q} \approx \frac{1}{p}$$

в предположении, что $p < q$. То, что p нельзя брать слишком маленьким, вытекает из алгоритма факторизации, обсуждаемого в следующем подразделе.

Ввиду того, что упомянутые выше "атаки" имеют такую малую вероятность успеха, мы не будем обсуждать их здесь. Но на них базируются некоторые из задач в конце главы.

9.2.2 Алгоритм факторизации для специального класса целых чисел

Сейчас мы кратко обсудим алгоритм факторизации, который работает быстрее, чем общие алгоритмы факторизации, рассматриваемые ниже, но в предположении, что хотя бы один из простых делителей числа n, скажем, p, обладает тем свойством, что $p - 1$ имеет только малые простые делители.

□ $(p-1)$-метод Полларда

Поллард в [Poll75] описал способ разложения числа n за $\sqrt{p}$ шагов, где p — наименьший простой делитель числа n. Это объясняет, почему нужно брать p и q большими.

Основным предположением в $(p-1)$-методе Полларда является то, что в разложении числа n хотя бы один из двух[2] делителей, скажем, p, обладает тем свойством, что $p-1$ имеет только малые простые делители. Чтобы быть точнее, назовем целое число *гладким* (см. также подразд. 8.3.4) относительно натурального числа[3] S, если все простые делители данного числа не превосходят S. Мы будем предполагать, что $p-1$ — гладкое относительно некоторого S.

Пример 9.3

Для простого числа $p = 70877$ число $p-1$ — гладкое относительно $S = 50$, что можно проверить с помощью функций FactorInteger и PrimeQ из пакета "Mathematica".

```
p = 70877; PrimeQ[p]
FactorInteger[p - 1]
```

```
True
```

```
{{2, 2}, {13, 1}, {29, 1}, {47, 1}}
```

Для каждого простого числа $r \leq S$ наибольшая степень r, которая все еще не превосходит n, удовлетворяет неравенству

$$r^i \leq n \qquad \text{или, эквивалентно,} \qquad i \leq \log_r n.$$

Определим R равенством

$$R = \prod_{r \leq S,\, r \text{ простое}} r^{\lfloor \log_r n \rfloor}. \tag{9.8}$$

Пример 9.4 (часть 1)

Рассмотрим число $n = 6700892281$ и предположим, что хотя бы для одного его простого делителя p число $p-1$ гладко относительно $S = 50$. Как вытекает из

```
Prime[15]
Prime[16]
```

```
47
```

```
53
```

[2] В действительности, составное n, к которому применим как этот алгоритм, так и излагаемые ниже, может иметь более двух простых делителей. — *Прим. ред.*

[3] Такое S часто называют *границей гладкости*. — *Прим. ред.*

имеются 15 *простых чисел, не превосходящих* $S = 50$. *С помощью функций* Prime, Log *и* Floor *из пакета "Mathematica" можно вычислить* R *по формуле* (9.8):

```
n = 6700892281; R = ∏_{i=1}^{15} (Prime[i])^Floor[Log[Prime[i], n]]
```

```
40495671803608715715481098873511450516871546389351490245060727076702214282424813734946501919403167962039754577870030089486336000000000000000
```

Посмотрим (любопытства ради) на показатели простых чисел, входящих в разложение R *и меньших* 50:

```
FactorInteger[R]
```

```
{{2, 32}, {3, 20}, {5, 14}, {7, 11}, {11, 9}, {13, 8},
{17, 7}, {19, 7}, {23, 7}, {29, 6}, {31, 6}, {37, 6},
{41, 6}, {43, 6}, {47, 5}}
```

Если $p-1$ — гладкое относительно S, то каждая степень r^i простого числа, которая делит $p-1$, будет также делителем числа R, так как i не превосходит $\lfloor \log_r n \rfloor$. Отсюда следует, что $p-1$ делит R. По теореме Ферма (теорема А.15) любое целое a, $1 \leq a < p$, удовлетворяет сравнению $a^{p-1} \equiv 1 \pmod{p}$. Поскольку $(p-1)|R$, также имеем $a^R \equiv 1 \pmod{p}$.

Возьмем теперь случайное целое a, $2 \leq a < p$, и проверим, будет ли НОД$(a, n) = 1$. Если этот НОД не равен 1, то мы нашли делитель числа n.

Если НОД$(a, n) = 1$, то из $a^R \equiv 1 \pmod{p}$ следует, что $p|(a^R - 1)$. Поскольку весьма неправдоподобно, что еще и $a^R \equiv 1 \pmod{q}$, мы почти наверняка находим делитель числа n (а именно p), вычисляя НОД$(a^R - -1, n)$. Заметим, что при этом не нужно вычислять a^R, — достаточно значения $a^R \bmod n$.

Пример 9.4 (часть 2)

Чтобы найти делитель числа $n = 6700892281$, *выберем случайное* a *между* 2 *и* $n-1$ *и вычислим* НОД$(a^R - 1, n)$ *посредством функций* Random, PowerMod *и* GCD *из пакета "Mathematica".*

```
a = Random[Integer, {2, n}]
GCD[PowerMod[a, R, n] - 1, n]
```

```
3922094384
```

```
81919
```

Это означает, что $p = 81919$ — *делитель числа* n. *Другим делителем будет* $n/p = 81799$. *Заметим, что если бы* q *тоже было гладким относительно* S, *то при вычислении* НОД *мы получили бы* n.

Резюмируем $(p-1)$-метод Полларда на следующем рисунке.

input: НАТУРАЛЬНОЕ ЧИСЛО n;
select ГРАНИЦУ ГЛАДКОСТИ S;
calculate R ИЗ (9.8);
select СЛУЧАЙНОЕ a, $2 \le a < n$;
calculate НОД$(a^R - 1, n)$;
if $1 < d < n$, **then** d — ДЕЛИТЕЛЬ n
else STOP ИЛИ ВЫБРАТЬ НОВОЕ СЛУЧАЙНОЕ a.

Рис. 9.1. $(p-1)$-метод факторизации Полларда.

Для того, чтобы $(p-1)$-метод Полларда оказался невыполнимым, при формировании системы RSA часто выбирают так называемые *безопасные простые числа*. Это простые числа p вида $p = 2p' + 1$, где p' — (большое) простое число. В этом случае $p - 1$ имеет ровно один малый делитель.

9.2.3 Общие алгоритмы факторизации

□ ρ-метод Полларда

Пусть p — неизвестный простой делитель натурального числа n, которое мы хотим разложить. Взглянем на последовательность $a_0, a_1, \ldots$, определяемую рекурсивными соотношениями

$$\begin{aligned} a_0 &= 1, \\ a_{i+1} &\equiv a_i^2 + 1 \pmod{p}, \qquad i \ge 0. \end{aligned}$$

Допустим, что мы нашли такие индексы u и v, $v > u$, что $a_u \equiv a_v \pmod{p}$. Тогда ясно, что НОД$(a_v - a_u, n)$ делится на p и весьма вероятно, что этот НОД равен p.

Конечно, p не известно, поэтому заменим прежнюю рекурсию следующей:

$$\begin{aligned} a_0 &= 1, \\ a_{i+1} &\equiv a_i^2 + 1 \pmod{n}, \qquad i \ge 0. \end{aligned} \tag{9.9}$$

Так как $p|n$, мы найдем делитель p из НОД$(a_v - a_u, n)$ для тех же самых значений u и v (вероятность того, что другие большие делители числа n делят этот НОД, ничтожно мала).

Вместо того, чтобы записывать все ранее вычисленные значения a_i, $i \ge 0$, воспользуемся алгоритмом Флойда нахождения цикла: именно, ищем такой индекс k, что $a_{2k} \equiv a_k$ и тогда берем $u = k$ и $v = 2k$. Идея проста: начиная с a_1 и a_2, рекурсивно определяем пару $(a_{i+1}, a_{2(i+1)})$, исходя из (a_i, a_{2i}).

Резюмируем сказанное выше на следующем рисунке.

input: НАТУРАЛЬНОЕ ЧИСЛО n;
put $a = 1, b = 2$;
do $a \leftarrow (a^2 + 1) \bmod n$;
$b \leftarrow \big(((b^2 + 1) \bmod n)^2 + 1\big) \bmod n$
until $d = \text{НОД}(a, b) > 1$;
if $d < n$, **then** d — ДЕЛИТЕЛЬ n
else STOP.

Рис. 9.2. ρ-метод Полларда факторизации n.

Пример 9.5

Чтобы найти делитель числа $n = 168149075693$ *рассмотренным выше методом, используем функции* While, Mod *и* GCD *пакета "Mathematica".*

```
n = 168149075693;
a = 1; b = 2; d = GCD[b - a, n];
While[d == 1, a = Mod[a^2 + 1, n];
  b = Mod[(Mod[b^2 + 1, n])^2 + 1, n]; d = GCD[b - a, n]];
d
```

```
350377
```

Итак, $p = 350377$ *— делитель числа* $n = 168149075693$. *Частным* n/p *служит* 479909, *которое тоже оказывается простым, что легко проверяется посредством функции* PrimeQ:

```
a = n / 350377
PrimeQ[a]
```

```
479909
```

```
True
```

□ Факторизация методом случайных квадратов

Как этот, так и последующий методы родственны методу исчисления индексов, изложенному в подразд. 8.3.4. Читатель может захотеть прочитать введение к тому методу, но для понимания следующего обсуждения это не обязательно. Мы предполагаем, что n — нечетное составное число.

Метод состоит из следующих четырех шагов.

Шаг 1:

Строим множество $S = \{p_1, p_2, \ldots, p_k\}$, состоящее из первых k простых чисел, т.е. $p_1 = 2, p_2 = 3$ и т.д. Множество S называется *базой делителей.*

Шаг 2:

Находим достаточно много пар (a_i, b_i), таких, что

$$a_i^2 \equiv b_i \pmod{n}, \tag{9.10}$$

причем b_i — гладкое относительно S, т.е. все b_i полностью разлагаются в произведение элементов базы делителей S, скажем,

$$b_i = \prod_{j=1}^{k} p_j^{u_{i,j}}, \quad \text{где} \quad u_{i,j} \geq 0.$$

Положим $\underline{u}_i = (u_{i,1}, u_{i,2}, \ldots, u_{i,k})$. Пары (a_i, b_i), удовлетворяющие сравнению (9.10), могут быть найдены попытками случайного выбора a_i. Альтернатива — использование подходящих рекурсивных соотношений, которые порождают кандидатов на роль a_i. Например, после испытания $a_i = a$ можно испытать $a_i = (a^2 + 1) \bmod n$.

Шаг 3:

Найдем семейство чисел b_i, произведение которых является квадратом. Вполне очевидно, что в этом условии имеет значение только четность показателей $u_{i,j}$. Поэтому положим $v_{i,j} = u_{i,j} \bmod 2$ и $\underline{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,k})$. Будем кратко писать $\underline{v}_i = \underline{u}_i \bmod 2$.

Так как любые $k + 1$ векторов $\underline{v}_i$ (длины k) будут линейно зависимы над $\mathbb{Z}_2$, должна существовать нетривиальная линейная комбинация, равная $\underline{0}$. Такая линейная комбинация может быть очень эффективно найдена стандартными методами линейной алгебры.

Пусть I — такое подмножество из $\{1, 2, \ldots, k\}$, что $\sum_{i \in I} \underline{v}_i \equiv \underline{0} \pmod 2$. Положим

$$x = \prod_{i \in I} a_i \qquad \text{и} \qquad y = \left(\prod_{i \in I} b_i\right)^{1/2}.$$

Шаг 4:

Из сравнения (9.10) следует, что $x^2 \equiv y^2 \pmod n$, т.е. n делит $(x - y)(x + y)$. Допустим, что $x \not\equiv \pm y \pmod n$ (вероятность этого, как мы вскоре увидим и как будет продемонстрировано в подразд. 9.5.1 для случая, когда n — произведение двух различных простых чисел, не меньше $1/2$). Тогда $x - y$ должно делиться на нетривиальный делитель числа n. Другими словами, НОД$(x - y, n)$ даст такой нетривиальный делитель.

Если НОД$(x - y, n) = n$, нужно попытаться найти другой квадрат — либо беря другую линейную зависимость между $\underline{v}_i$, либо заменяя одну из пар (a_i, b_i) новой парой.

Рассмотрим сравнение $x^2 \equiv y^2 \pmod n$, где преполагается, что y имеет заданное фиксированное значение, взаимно простое с n. Далее, пусть p^a — любой делитель в разложении n на степени простых чисел (см. теорему А.6). Тогда сравнение $x^2 \equiv y^2 \pmod{p^a}$ имеет в точности два решения, а именно $x \equiv \pm y \pmod{p^a}$. В самом деле, при $a = 1$ это следует из теоремы В.15. При $a > 1$ все еще верно, что p^a должно делить либо $x - y$, либо $x + y$, но не оба числа, потому что если $p | (x - y)$ и $p | (x + y)$, то $p | 2y$, причем $p \nmid y$, а так как n нечетно, то и p нечетно. Получаем[4], что $x \equiv \pm y \pmod{p^a}$ и при $a > 1$.

[4] Можно проще: $\mathbb{Z}_{p^a}^*$ — циклическая группа, значит, для любого d, делящего ее порядок, в группе существуют ровно $\varphi(d)$ элементов порядка d. Здесь $d = 2$, $\varphi(d) = 1$.— *Прим. ред.*

Теперь непосредственно из китайской теоремы об остатках (теорема A.19) вытекает, что сравнение $x^2 \equiv y^2 \pmod{n}$ имеет 2^l решений, где l — число различных простых чисел, делящих n. Только два из этих 2^l, $l \geq 2$, решений имеют вид $x \equiv \pm y \pmod{p^a}$. Поэтому вероятность того, что НОД$(x-y, n)$ дает нетривиальный делитель числа n, не меньше, чем $(2^l - 2)/2^l \geq 2/4 = 1/2$.

input: НАТУРАЛЬНОЕ ЧИСЛО n;
make БАЗУ ДЕЛИТЕЛЕЙ $S = \{p_1, \ldots, p_k\}$;
find ПАРЫ (a_i, b_i), ГДЕ a_i СЛУЧАЙНО,
$a_i^2 \equiv b_i \pmod{n}$, b_i ГЛАДКО ОТНОСИТЕЛЬНО S;
find ТАКОЕ ИНДЕКСНОЕ МНОЖЕСТВО I,
ЧТО $\prod_{i \in I} b_i$ — КВАДРАТ;
put $x = \prod_{i \in I} a_i, y = \sqrt{\prod_{i \in I} b_i}$;
put $d =$ НОД$(x - y, n)$;
if $d < n$, **then** d — ДЕЛИТЕЛЬ n
else ПОВТОРИТЬ ПОПЫТКУ С ДРУГИМ I.

Рис. 9.3. Метод случайных квадратов.

Пример 9.6

Попробуем разложить число $n = 1271$ рассмотренным выше методом. Сначала, пользуясь функциями Table и Prime пакета "Mathematica", построим базу делителей из первых 8 простых чисел.

```
S = Table[Prime[i], {i, 1, 8}]
```

```
{2, 3, 5, 7, 11, 13, 17, 19}
```

Теперь используем функцию Random для генерации случайного a, $1 \leq a \leq n$, и функцию FactorInteger для разложения числа $b \equiv a^2 \pmod{n}$.

```
n = 1271; a = Random[Integer, {1, n}]
b = Mod[a^2, n]
FactorInteger[b]
```

```
460
```

```
614
```

```
{{2, 1}, {307, 1}}
```

К сожалению, $b = 614$ не гладко относительно S, но после нескольких попыток и ошибок мы находим следующие девять гладких чисел (составляющих список, названный a).

```
n = 1271;
a = {583, 536, 1137, 421, 727, 1034, 1051, 107, 1111};
```

```
b = Mod[a^2, n];
TableForm[    Table[{a[[i]], b[[i]],
    FactorInteger[b[[i]] ] // OutputForm},
    {i, 1, Length[a]}], TableHeadingth ->
    {{}, {"a", "\!\(a\^2\) mod n", "factors"}},
TableAlignments -> {Left}]
```

a	a^2 mod n	factors
583	532	{{2, 2}, {7, 1}, {19, 1}}
536	50	{{2, 1}, {5, 2}}
1137	162	{{2, 1}, {3, 4}}
421	572	{{2, 2}, {11, 1}, {13, 1}}
727	1064	{{2, 3}, {7, 1}, {19, 1}}
1034	245	{{5, 1}, {7, 2}}
1051	102	{{2, 1}, {3, 1}, {17, 1}}
107	10	{{2, 1}, {5, 1}}
1111	180	{{2, 2}, {3, 2}, {5, 1}}

Показатели в разложениях чисел b_i задаются векторами $\underline{u}_i$, которые образуют строки матрицы U ниже. Векторы $\underline{v}_i$ — это $\underline{u}_i$, приведенные по модулю 2. Они образуют строки матрицы V.

Например, $b_1 = 532 = 2^2 \cdot 7 \cdot 19$ дает $\underline{u}_1 = \{2,0,0,1,0,0,0,1\}$ и $\underline{v}_1 = \{0,0,0,1,0,0,0,1\}$. Эти две строки суть первые строки матриц U и V соответственно. Чтобы показать их, воспользуемся функцией MatrixForm.

```
U = {{2, 0, 0, 1, 0, 0, 0, 1},    {1, 0, 2, 0, 0, 0, 0, 1},
    {1, 4, 0, 0, 0, 0, 0, 0},    {2, 0, 0, 0, 1, 1, 0, 0},
    {3, 0, 0, 1, 0, 0, 0, 1},    {0, 0, 1, 2, 0, 0, 0, 0},
    {1, 1, 0, 0, 0, 0, 1, 0},    {1, 0, 1, 0, 0, 0, 0, 0},
    {2, 2, 1, 0, 0, 0, 0, 0}};
V = Mod[U, 2];
MatrixForm[U]
MatrixForm[V]
```

$$\begin{pmatrix} 2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\ 1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

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

Чтобы найти нетривиальную линейную комбинацию строк матрицы V, сравнимую с нулевым вектором по модулю 2, используем функции NullSpace и Transpose.

```
NullSpace[Transpose[V], Modulus -> 2]
```

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

Видно, что первая из найденных линейных зависимостей между строками матрицы V отражает идентичность двух строк, но третья зависимость дает пригодное к использованию индексное множество I, а именно, $I = \{1, 3, 5\}$. Это приводит к значениям $x = a_1 a_3 a_5$ и $y = \sqrt{b_1 b_3 b_5}$.

```
x = a[[1]] * a[[3]] * a[[5]]
y = Sqrt[b[[1]] * b[[3]] * b[[5]]]
GCD[x - y, n]
```

```
481907217
```

```
9576
```

```
41
```

Таким образом, $p = 41$ — делитель числа $n = 1271$. И в самом деле, $1271 = 31 \times 41$:

```
n / 41
```

```
31
```

□ Квадратичное решето

Сложность этого метода —

$$\exp\{1.923\ldots (\ln n)^{1/3} (\ln \ln n)^{2/3}\} \qquad \text{операций.}$$

Как и в случае предыдущих методов, мы не будем объяснять все детали этой техники факторизации.

Для начала мы нуждаемся в так называемой базе делителей — списке чисел, которые будут определены позже.

Пусть $r = \lfloor \sqrt{n} \rfloor$, а многочлен $f(x)$ определяется равенством

$$f(x) = (x+r)^2 - n = x^2 + 2r \cdot x + r^2 - n.$$

Заметим, что $r^2 \le n < (r+1)^2$ и, значит, $0 \le n - r^2 < 2r + 1 \le 2\sqrt{n} + 1$. Отсюда следует, что если x мало по абсолютной величине, то и $f(x)$ будет мало (по сравнению с n).

Для $x = 0, \pm 1, \pm 2, \ldots$ определим $a = x + r$ и проверим число $b = (x+r)^2 - n$ на гладкость относительно S, т.е. проверим, все ли простые делители числа b лежат в S. Если это так, занесем пару (a, b) в список пар (a_i, b_i), обладающих этим свойством. Заметим, что $a_i^2 = (x+r)^2 \equiv b_i \pmod{n}$, — точно как в сравнении (9.10).

Если простое число p делит b_i, то $p|((x+r)^2 - n)$ для некоторого известного значения x. Это означает, что $n \equiv (x+r)^2 \pmod{p}$, и поэтому n — квадратичный вычет по модулю p. Значит, только простые делители, появляющиеся в разложении каких-либо b_i, дают символ Якоби $(n/p) = 1$.

Таким образом, составим базу делителей S из k наименьших нечетных простых чисел p_j, $1 \le j \le k$, обладающих тем свойством, что $(n/p_j) = 1$. Добавим к S также -1 и 2, потому что b_i могут быть отрицательными или четными.

Теперь, зная, как строить список пар (a_i, b_i), удовлетворяющих условиям

$$a_i^2 \equiv b_i \pmod{n},$$
$$b_i \text{ — гладкое относительно } S,$$

мы можем продолжить, начиная с шага 3, алгоритм, описанный в предыдущем подразделе.

Резюмируем метод квадратичного решета на следующем рисунке.

input: НАТУРАЛЬНОЕ ЧИСЛО n;
make БАЗУ ДЕЛИТЕЛЕЙ $S = \{-1, 2, p_1, \ldots, p_k\}$,
ГДЕ $(n/p_j) = 1$;
find ПАРЫ (a_i, b_i), ГДЕ $a_i - \lfloor \sqrt{n} \rfloor$ МАЛО,
$a_i^2 \equiv b_i \pmod{n}$ И b_i ГЛАДКО ОТНОСИТЕЛЬНО S;
find ИНДЕКСНОЕ МНОЖЕСТВО I, ТАКОЕ,
ЧТО $\prod_{i \in I} b_i$ — КВАДРАТ;
put $x = \prod_{i \in I} a_i, y = \sqrt{\prod_{i \in I} b_i}$;
put $d =$ НОД$(x - y, n)$;
if $d < n$, **then** d — ДЕЛИТЕЛЬ ЧИСЛА n
else ПОВТОРИТЬ ПОПЫТКУ С ДРУГИМ I.

Рис. 9.4. Факторизация методом квадратичного решета.

Мы дадим пример только первых двух шагов метода квадратичного решета.

Пример 9.7

Пусть $n = 661643$. *Для создания базы делителей с* 10 *нечетными простыми числами воспользуемся функциями* While, Length, JacobiSymbol, Prime *и* AppendTo *пакета "Mathematica".*

```
n = 661643; k = 10;
SS = {-1, 2}; i = 2;
While[Length[SS] - 2 < k,
   If[JacobiSymbol[n, Prime[i]] == 1,
   AppendTo[SS, Prime[i]]]; i = i + 1];
SS
```

```
{-1, 2, 11, 19, 23, 31, 37, 47, 53, 59, 79, 89}
```

Чтобы узнать, будут ли какие-то из чисел $f(-5), f(-4), \ldots, f(5)$ *гладкими относительно* S, *используем функции* TableForm, Table *и* FactorInteger:

```
n = 661643; Clear[x, f];
r = ⌊√n⌋; m = 5;
f(x_) := (x + r)^2 - n;
TableForm[    Table[{r + i, f[i],
   FactorInteger[f[i]] // OutputForm},    {i, -m, m}]]
```

```
808    -8779    {{-1, 1}, {8779, 1}}
809    -7162    {{-1, 1}, {2, 1}, {3581, 1}}
810    -5543    {{-1, 1}, {23, 1}, {241, 1}}
811    -3922    {{-1, 1}, {2, 1}, {37, 1}, {53, 1}}
812    -2299    {{-1, 1}, {11, 2}, {19, 1}}
813    -674     {{-1, 1}, {2, 1}, {337, 1}}
814    953      {{953}}
815    2582     {{2, 1}, {1291, 1}}
816    4213     {{11, 1}, {383, 1}}
817    5846     {{2, 1}, {37, 1}, {79, 1}}
818    7481     {{7481, 1}}
```

Видно, что мы нашли только три подходящие пары (a_i, b_i), *а именно* $(811, -3922)$, $(812, -2299)$ *и* $(817, 5846)$. *Таким образом, нужно попробовать расширить область значений. Завершение этого примера мы оставим читателю в качестве упражнения (см. задачу* 9.7*).*

9.3 Некоторые опасные режимы RSA

9.3.1 Малый публичный показатель

Мы обсудим здесь две частные опасности, описанные в [Håst88] (см. также [CoppFPR96][5]). Первую опасность создает ситуация, когда несколько персон выбрали один и тот же (малый) показатель, а отправитель хочет передать им всем одно и то же сообщение. Вторая опасность возникает, когда отправитель хочет передать несколько математически связанных сообщений одному получателю, имеющему малый публичный показатель.

Обе опасности могут показаться читателю надуманными, но, поскольку возведение в степень по модулю больших чисел оказывается довольно обременительным, в практических ситуациях выбор малых публичных показателей остается весьма привлекательным.

□ Передача одинаковых сообщений нескольким получателям, имеющим один и тот же малый публичный ключ

Предположим, что Алиса хочет послать одно и то же секретное сообщение Бобу, Чаку и Деннису. Пусть публичные модули этих трех получателей равны n_B, n_C и n_D соответственно. Допустим теперь, что у всех троих оказался один и тот же публичный показатель $e = 3$. Сообщения, передаваемые Алисой, суть

$$\begin{array}{ll} c_B = m^3 \bmod n_B & \text{для Боба,} \\ c_C = m^3 \bmod n_C & \text{для Чака,} \\ c_D = m^3 \bmod n_D & \text{для Денниса.} \end{array} \tag{9.11}$$

Почти наверняка все три модуля будут попарно взаимно просты (в противном случае по меньшей мере два из них подвергаются риску). Злоумышленница Ева, перехватившая c_B, c_C и c_D, может использовать китайскую терему об остатках (теорема A.19), чтобы из (9.11) найти $m^3 \bmod n_B n_C n_D$. Поскольку можно считать, что $m < \min\{n_B, n_C, n_D\}$, выполняется неравенство $m^3 < n_B n_C n_D$. Это означает, что фактически Ева находит целое число m^3. Вычислить m не составляет труда.

Пример 9.8

Предположим, что $n_B = 137703491$, $n_C = 144660611$ *и* $n_D = 149897933$, *а* $c_B = 124100785$, $c_C = 85594143$ *и* $c_D = 148609330$ — *три перехваченных сообщения.*

Чтобы решить систему линейных сравнений

$$m^3 \equiv c_B \pmod{n_B},\ m^3 \equiv c_C \pmod{n_C},\ m^3 \equiv c_D \pmod{n_D},$$

[5] А также полезный обзор *[Bone99] теоретических атак на RSA, включающий доказательства ряда результатов.— *Прим. ред.*

с известными правыми частями и известными модулями, воспользуемся функцией ChineseRemainderTheorem из "Mathematica". С этой целью загружаем сначала пакет NumberTheory'NumberTheoryFunctions'.

```
<< NumberTheory`NumberTheoryFunctions`
```

```
nB = 137703491; nC = 144660611; nD = 149897933;
cB = 124100785; cC = 85594143; cD = 148609330;
mCubed = ChineseRemainderTheorem[{cB, cC, cD}, {nB, nC, nD}]
```

```
|| 1881563525396008211918161
```

Получаем, что $m^3 \equiv 1881563525396008211918161 \pmod{n_B n_C n_D}$. *Поскольку* $m^3 < n_B n_C n_D$, *мы даже имеем*

$$m^3 = 1881563525396008211918161.$$

Теперь легко найти m:

```
m = (mCubed)^(1/3)
```

```
|| 123454321
```

Правильность этого выхода легко проверяется посредством функции Mod.

```
Mod[m^3, nB] == cB
Mod[m^3, nC] == cC
Mod[m^3, nD] == cD
```

```
|| True

|| True

|| True
```

□ Передача связанных сообщений получателю с малым публичным ключом

Алиса хочет послать два секретных сообщения Бобу, публичный показатель $e = e_B$ которого оказался довольно маленьким. Пусть n_B — модуль Боба. Допустим теперь, что эти два сообщения связаны линейно, т.е. $m_2 = a \cdot m_1 + b$, где a и b лежат в $\mathbb{Z}_{n_B}$, и что перехватчице Еве известно это линейное соотношение.

Копперсмит и др. описывают в [CoppFPR96] два неожиданных метода, позволяющих Еве восстановить открытый текст m_1.

Прямой метод

Сначала опишем этот метод для случая $e = 3$. Пусть шифровки сообщений m_1 и m_2 обозначаются посредством c_1 и c_2 соответственно. Таким

образом, $c_1 = m_1^3 \bmod n_B$ и $c_2 = (a \cdot m_1 + b)^3 \bmod n_B$. Тогда

$$\frac{b(c_2 + 2a^3c_1 - b^3)}{a(c_2 - a^3c_1 + 2b^3)} \equiv \frac{3a^3bm_1^3 + 3a^2b^2m_1^2 + 3ab^3m_1}{3a^3bm_1^2 + 3a^2b^2m_1 + 3ab^3} \equiv m_1 \pmod{n_B}. \tag{9.12}$$

Эти вычисления можно проверить с помощью функции *Simplify* из пакета "Mathematica" следующим образом:

```
Clear[a, b, c1, c2, m1, m2];
Simplify[
 b (c2 + 2 a^3 c1 - b^3)
 ------------------------ //.{c1 -> m1^3,
 a (c2 - a^3 c1 + 2 b^3)
        c2 -> (a * m1 + b)^3}]
```

```
|| m1
```

Отметим особенно простой случай, когда $m_1 = m$ и $m_2 = m + 1$ т.е. $a = b = 1$. Тогда соотношение (9.12) сводится к сравнению

$$\frac{(m+1)^3 + 2m^3 - 1}{(m+1)^3 - m^3 + 2} \equiv \frac{3m^3 + 3m^2 + 3m}{3m^2 + 3m + 3} \equiv m \pmod{n_B}.$$

Пример 9.9

Предположим, что $n_B = 477310661$ *и что сообщения* m_1 *и* m_2 *связаны сравнением* $m_2 \equiv 3m_1 + 5 \pmod{n_B}$. *Таким образом,* $a = 3$ *и* $b = 5$. *Пусть* $c_1 = 5908795$, $c_2 = 374480016$. *Тогда* m_1 *можно вычислить в пакете "Mathematica" с помощью функций* *Mod* *и* *Solve* *следующим образом:*

```
Clear[c1, c2, f, g, m1, m2, a, b];
n = 477310661;
c1 = 5908795; c2 = 374480016;
a = 3; b = 5;
f = Mod[b (c2 + 2 a^3 c1 - b^3), n];
g = Mod[a (c2 - a^3 c1 + 2 b^2), n];
Solve[{f == g * m1, Modulus == n}, m1]
```

```
|| {{Modulus -> 477310661, m1 -> 321321321}}
```

Итак, найдено $m_1 = 321321321$. *Легко проверить, что это действительно искомое решение:*

```
m1 = 321321321;
m2 = Mod[3 * m1 + 5, n]
PowerMod[m1, 3, n] == c1
PowerMod[m2, 3, n] == c2
```

```
|| 9342646

|| True

|| True
```

Если $a = b = 1$ и $e = e_B > 3$, тот же метод все еще работает. Фактически можно показать ([CoppFPR96]), что существуют многочлены $P(m)$ и $Q(m)$, такие, что каждый из них выражается в виде многочлена от $c_1 = m^e \bmod n_B$ и $c_2 = (m+1)^e \bmod n_B$, причем $Q(m) = m \cdot P(m)$. Приведем такие многочлены для $e = 5$:

$$\begin{aligned} P(m) &= c_2^3 + 2c_1c_2^2 - 4c_1^2c_2 + c_1^3 - 2c_2^2 + 9c_1c_2 + 8c_1^2 + c_2 - 2c_1, \\ Q(m) &= 9c_1c_2^2 - 9c_1^2 \end{aligned}$$

Опять можно проверить сказанное:

```
Clear[c1, c2, m];
P = c2^3 + 2 c1 c2^2 - 4 c1^2 c2 + c1^3 - 2 c2^2 +
  9 c1 c2 + 8 c1^2 + c2 - 2 c1; Q = 9 c1 c2^2 - 9 c1^2;
Expand[P //. {c1 -> m^3, c2 -> (m + 1)^3}]
Expand[Q //. {c1 -> m^3, c2 -> (m + 1)^3}]
Simplify[Q/P //. {c1 -> m^3, c2 -> (m + 1)^3}]
```

```
|| 9m^2 + 54m^3 + 135m^4 + 171m^5 + 135m^6 + 54m^7 + 9m^8

|| 9m^3 + 54m^4 + 135m^5 + 171m^6 + 135m^7 + 54m^8 + 9m^9

|| m
```

Чтобы найти решение в общем случае, запишем $P = \sum_{i+j \le e} p_{i,j} c_2^i c_1^j$ и $Q = \sum_{i+j \le e} q_{i,j} c_2^i c_1^j$. Теперь подставим $c_2 = (m+1)^e$ и $c_1 = m^e$ в P и Q и получим многочлены от m степени $\le e^2$. Затем приравняем соответствующие коэффициенты, добиваясь равенства $Q(m) = m \cdot P(m)$. Это даст $2((e+1) + e + \ldots + 2 + 1) = 2\binom{e+2}{2} = (e+2)(e+1)$ линейных уравнений относительно коэффициентов в P и Q. Таким образом, фактически имеется большое пространство решений.

Так как количество членов в $P(m)$ и $Q(m)$ растет квадратично относительно e, изложенный подход становится довольно трудоемким при бóльших значениях e.

Метод взлома вычислением НОД

Для произвольных значений e существует еще более прямой путь определения m_1 и m_2 из c_1 и c_2 при условии, что сообщения удовлетворяют полиномиальному сравнению, которое известно перехватчику. Предположим, что $m_2 \equiv f(m_1) \pmod{n_B}$. Идея состоит в вычислении НОД$(z^e - c_1, (f(z))^e - c_2)$. В самом деле, поскольку m_1 — нуль обоих многочленов, они делятся на $z - m_1$. Как следствие, НОД также

имеет этот делитель, и почти наверняка у него нет никаких других делителей. Продемонстрируем эту идею на примере.

Пример 9.10

Пусть $e = 5, n_B = 466883$. *Предположим, что сообщения* m_1 *и* m_2 *связаны равенством* $m_2 = 2m_1 + 3$, *а их шифртекстами служат* $c_1 = 66575$ *и* $c_2 = 387933$ *соответственно. Мы хотим вычислить* НОД$(z^5 - 66575, (2z+3)^5 - 387933) \bmod 466883$. *Вообще говоря, "Mathematica" не может сделать это непосредственно, так как* n_B *— составное число. Поэтому мы просто, шаг за шагом, последуем полиномиальной версии алгоритма Эвклида. При этом могут возникать проблемы, когда появляются числа, не взаимно простые с* $n = n_B$. *Такое встречается редко и, кроме того, это совсем не плохо. В самом деле, на этом пути почти всегда отыщется нетривиальный делитель числа* n, *так что система окажется взломанной!*

Сначала мы вычисляем $f_1 = (2z+3)^5 - 387993$ *и* $f_2 = z^5 - 66575$, *а затем делим* f_1 *на* f_2. *Воспользуемся функциями* Expand *и* PolynomialMod *пакета "Mathematica".*

```
n = 466883;
c1 = 66575; c2 = 387933;
f1 = Expand[(2 z + 3)^5 - c2]
f2 = z^5 - c1
f3 = PolynomialMod[f1 - 32 f2, n]
```

```
|| -387690 + 810z + 1080z^2 + 720z^3 + 240z^4 + 32z^5
|| -66575 + z^5
|| 342061 + 810z + 1080z^2 + 720z^3 + 240z^4
```

Чтобы сделать процесс деления более удобным, нормализуем f_3, *умножая его на мультипликативный обратный (по модулю* n_B*) к старшему коэффициенту с помощью функции* PowerMod *из пакета "Mathematica".*

```
InverseLeadCoeff = PowerMod[240, -1, n]
f3 = PolynomialMod[InverseLeadCoeff * f3, n]
```

```
|| 258731
|| 376877 + 408526z + 233446z^2 + 3z^3 + z^4
```

Продолжаем этот процесс деления, пока для некоторого k *не будет* $f_k = 0$. *Тогда* f_{k-1} *— искомый* НОД.

```
f4 = PolynomialMod[f2 - f3 * (z + 466880), n]
```

```
|| 130290 + 381818z + 291812z^2 + 233446z^3
```

```
InverseLeadCoeff = PowerMod[233446, -1, n]
f4 = PolynomialMod[InverseLeadCoeff * f4, n]
```

```
103752
```

```
184581 + 292352z + 116723z^2 + z^3
```

```
f5 = PolynomialMod[f3 - f4 * (z + 350163), n]
```

```
355162 + 4681z + 203714z^2
```

```
InverseLeadCoeff = PowerMod[203714, -1, n]
f5 = PolynomialMod[InverseLeadCoeff * f5, n]
```

```
349909
```

```
397084 + 98465z + z^2
```

```
f6 = PolynomialMod[f4 - f5 * (z + 18258), n]
```

```
451016 + 87731z
```

```
InverseLeadCoeff = PowerMod[87731, -1, n]
f6 = PolynomialMod[InverseLeadCoeff * f6, n]
```

```
132235
```

```
466340 + z
```

```
f7 = PolynomialMod[f5 - f6 * (z + 99008), n]
```

```
0
```

Мы получили, что $k = 7$ *и, следовательно,*

$$\text{НОД}(z^5 - 66575, (2z+3)^5 - 387933) \equiv z + 466340 \equiv z - 543 \pmod{466883}.$$

Поэтому секретным сообщением является $m = 543$. *Проверим это с помощью функции* PowerMod *из пакета "Mathematica".*

```
m = 543;
PowerMod[m, 5, n] == c1
PowerMod[2 m + 3, 5, n] == c2
```

```
True
```

```
True
```

Рассмотренный подход поиска m посредством вычисления НОД остается практичным для числа e длиной до 32 битов ([CoppFPR96]).

9.3.2 Малый секретный показатель; атака Винера

Винер в [Wien90] показал, насколько опасно использовать систему RSA с малым секретным показателем, где "малый" означает что-то наподобие $\sqrt{n}$. Это наблюдение важно, поскольку люди часто склонны выбирать малый показатель, чтобы уменьшить объем работы при возведении в степень. Например, когда smart-карта используется для подписывания сообщений (см. подразд. 9.1.3), она должна вычислять степени $c^d \bmod n$. Если вычислительная мощность карты ограничена, удобно взять относительно малое значение d (конечно, не настолько малое, чтобы d можно было найти исчерпывающим поиском).

Покажем сначала, что сравнение (9.4) можно заменить несколько более сильным сравнением

$$e \cdot d \equiv 1 \pmod{\text{НОК}[p-1, q-1]},$$

где НОК обозначает наименьшее общее кратное. Числа $p-1$ и $q-1$ оба делят $\varphi(n)$, а потому делят и $\text{НОК}[p-1, q-1]$. Теперь заметим, что для правильного функционирования системы RSA нужны лишь сравнения $e \cdot d \equiv 1 \pmod{p-1}$ и $e \cdot d \equiv 1 \pmod{q-1}$. Причина в том, что этих двух сравнений достаточно для доказательства формул (9.5) и (9.6) по модулю p и по модулю q соответственно. Тогда китайская теорема об остатках влечет, что сравнения (9.5) и (9.6) выполняются также по модулю n. Мы приходим к выводу, что достаточно сравнения $e \cdot d \equiv 1 \pmod{\text{НОК}[p-1, q-1]}$.

Последующий криптоанализ имеет дело с этим более сильным сравнением. Цель криптоаналитика — найти d, удовлетворяющее последнему соотношению (а также p и q). Наше сравнение можно переписать в виде

$$e \cdot d = 1 + K \cdot \text{НОК}[p-1, q-1] = 1 + \frac{K}{G}(p-1)(q-1),$$

где $G = \text{НОД}(p-1, q-1)$. Если K и G имеют общий нетривиальный делитель, то равенство выше упрощается до

$$e \cdot d = 1 + \frac{k}{g}(p-1)(q-1), \quad \text{где } \text{НОД}(k, g) = 1. \tag{9.13}$$

Нужно представлять себе, что G (а потому и g) очень мало. В типичной системе RSA числа p и q — безопасные простые, т.е. $p-1 = 2p'$ и $q-1 = 2q'$ с простыми числами p' и q', а в этом случае $G = 2$ и $g = 1$ или 2.

Перепишем равенство (9.13), разделив обе его части на $d \cdot n\,(= d \cdot p \cdot q)$ и перегруппировав члены:

$$\frac{k}{d \cdot g} = \frac{e}{n} + \frac{k}{d \cdot g}\left(\frac{1}{p} + \frac{1}{q} - \frac{1}{n}\right) - \frac{1}{d \cdot n}. \tag{9.14}$$

Мы хотим показать, что $k/(d \cdot g)$ — подходящая дробь к непрерывной дроби известного рационального числа e/n. Так как непрерывные дроби

легко вычисляются, можно найти секретный показатель d (а также k и g).

Теорема 9.2
Предположим, что $p \sim q \sim \sqrt{n}$ и $2g < d$. Тогда $k \sim d \cdot g$ и числа d, k, g, p и q могут быть найдены из непрерывной дроби числа e/n для секретного показателя $d \leq n^{1/4}$.

Замечание 1. Мы будем несколько вольно обращаться с символом $\sim$. Запись $a \sim b$ означает что-то вроде "a и b имеют одинаковый порядок по величине".

Замечание 2. Мы уже обсуждали правдоподобность того, что g мало. Если d выбирается как малое число, то e ведет себя подобно случайному числу из области $\{1, 2, \ldots, \text{НОК}[p-1, q-1]\}$, поэтому предположение $e \sim n$ вполне разумно. То же самое сохраняется и для соотношений $p \sim q \sim \sqrt{n}$ (см. обсуждение до и после примера 9.2).

Замечание 3. Равенство (9.14) влечет $\frac{k}{d \cdot g} > \frac{e}{n}$, следовательно, достаточно проверять только подходящие к e/n дроби нечетного порядка.

Доказательство теоремы 9.2. Если $e \sim n$, то в силу (9.14) $k \sim d \cdot g$, поскольку все другие члены там близки к нулю. Далее, из соотношения (9.14) вытекает, что

$$\left|\frac{k}{d \cdot g} - \frac{e}{n}\right| = \left|\frac{k}{d \cdot g}\left(\frac{1}{p} + \frac{1}{q} - \frac{1}{n}\right) - \frac{1}{d \cdot n}\right| \leq$$

$$\leq \frac{k+g}{d \cdot g \cdot n} + \frac{k}{d \cdot g}\left(\frac{1}{p} + \frac{1}{q}\right) \sim \frac{d+1}{d \cdot n} + \frac{2}{\sqrt{n}} \sim \frac{1}{\sqrt{n}}.$$

Так как $2g \cdot d < d^2 \leq n^{1/2}$, мы получаем

$$\left|\frac{k}{d \cdot g} - \frac{e}{n}\right| \leq \frac{1}{\sqrt{n}} < \frac{1}{2(d \cdot g)^2}.$$

Из теоремы А.35 следует, что число $k/(d \cdot g)$ равно подходящей дроби к непрерывной дроби числа e/n. Поскольку, в силу соотношения (9.13), $\text{НОД}(k, g) = \text{НОД}(k, d) = 1$, следствие А.32 влечет, что $k/(d \cdot g)$ — в точности (несократимая) подходящая дробь. Так как g очень мало, поиск g и d методом проб и ошибок требует небольших усилий[6].

[6] Последнее двойное неравенство в доказательстве нам не ясно. Покажем иначе, как найти d и $\varphi(n)$ (а значит, p и q), если выполнены условия: (1) $p < q < 2p$; (2) $d < (1/3)n^{1/4}$ ([Wien90]).

Для некоторого r имеем $ed - r\varphi(n) = 1$, где $\varphi(n) = n - p - q + 1$ и, значит, $n - \varphi(n) = p + q - 1 < 3p < 3\sqrt{n}$. Поэтому

$$\left|\frac{e}{n} - \frac{r}{d}\right| = \frac{|ed - r\varphi(n) + r\varphi(n) - rn|}{dn} = \frac{|1 + r(\varphi(n) - n)|}{dn} \leq \frac{3rn^{1/2}}{dn} = \frac{3r}{dn^{1/2}}.$$

Теперь из (9.13) можно вычислить $(p-1)(q-1)$ и, так как $p \cdot q$ известно, можно также найти разложение n на p и q.

■

Пример 9.11

Рассмотрим $n = 9998000099$ *и* $e = 6203014673$. *Вычислим последовательные подходящие дроби к* e/n. *Сначала загрузим в "Mathematica" пакет NumberTheory'ContinuedFractions', а затем используем функции ContinuedFraction и Normal.*

```
<< NumberTheory`ContinuedFractions`
```

```
n = 9998000099; e = 6203014673;
Normal[ContinuedFraction[e / n, 2]]
Normal[ContinuedFraction[e / n, 4]]
Normal[ContinuedFraction[e / n, 6]]
Normal[ContinuedFraction[e / n, 8]]
```

```
1        2/3        5/8        18/29
```

Выясним, почему последняя из этих дробей не приводит к d *(прочие случаи еще проще). Приравнивание* $k/(d \cdot g) = 18/29$ *дает значения* $k = 18$, $g = 1$ *и* $d = 29$. *Легкий путь демонстрации того, что это не является правильным значением* d, — *зашифровать результат дешифрования и убедиться, что ответ отличен от исходного сообщения. Воспользуемся функцией PowerMod:*

```
m = 123; d = 29;
c = PowerMod[m, e, n];
PopwerMod[c, d, n] == m
```

```
False
```

Испытаем следующую подходящую дробь:

```
Normal[ContinuedFraction[e / n, 10]]
```

```
85/137
```

Приравнивание $85/137 = k/(d \cdot g)$ *приводит к значениям* $k = 85$, $g = 1$ *и* $d = 137$. *Из (9.13) получаем* $(p-1)(q-1) = 9993745862$.

Но $r\varphi(n) = ed - 1 < ed < \varphi(n)d$, откуда $r < d < (1/3)n^{1/4}$. Следовательно,

$$\left|\frac{e}{n} - \frac{r}{d}\right| < \frac{n^{1/4}}{dn^{1/2}} = \frac{1}{dn^{1/4}} < \frac{1}{2d^2}.$$

Так как НОД$(r, d) = 1$, по теореме А.35 r/d — подходящая к e/n дробь. Дальше все ясно.— *Прим. ред.*

```
k = 85; g = 1; d = 137;
(e * d - 1) * g / k
```

```
9997800120
```

Учитывая, что $n = p \cdot q = 9998000099$, *получаем* $p + q - 1 = p \cdot q - (p-1)(q-1) =$

```
n - (e * d - 1) g / k
```

```
199979
```

Таким образом, p *и* q *— корни многочлена* $(x-p)(x-q) = x^2 - 199980x + 9998000099$. *Они могут быть найдены с помощью функции* Solve:

```
Clear[x];
Solve[x^2 - 199980 x + 9998000099 x^2 == 0, {x}]
```

```
{{x -> 99989}, {x -> 99991}}
```

В самом деле, $99989 \times 99991 = n$:

```
99989 * 99991 == n
```

```
True
```

9.3.3 Некоторые физические атаки

Физические атаки на криптографические устройства лежат за пределами этого введения. Несмотря на это, мы кратко упомянем две такие атаки ввиду их связи с излагаемой здесь теорией.

□ Атака по времени вычислений

Предположим, что система RSA реализована на электронном устройстве (наподобие smart-карты) и что секретное возведение в степень ($m \mapsto m^d \bmod n$ или $c \mapsto c^d \bmod n$) в процессе функционирования RSA следует вычислительной схеме типа объясненной в подразд. 8.1.1, т.е. любой схеме, которая состоит из повторяющихся возведений в квадрат и умножений (см. пример 8.3).

Далее, допустим, что при этой атаке (см. [Koch96]) наблюдатель может измерять электромагнитное излучение или потребление мощности устройством и тем самым может оценивать длительность различных вычислений. Обычно умножение требует больше времени, чем простое возведение в квадрат.

На этом пути атакующий может определить частную последовательность возведений в квадрат и умножений, через которую проходит программа. Основываясь на выходе, он может легко вычислить секретный показатель d, записанный на карте.

Например, если измерения дают Sq.Sq.M.Sq.Sq.M.Sq.Sq.M.Sq.M, где Sq означает возведение в квадрат, а M — умножение, то показатель получается, исходя из следующей процедуры:

```
Clear[a];
```

$$\left(\left(\left(\left(\left((a^2)^2 a\right)^2\right)^2 a\right)^2 a\right)^2 a\right)^2 a$$

$$\| \quad a^{171}$$

□ Атака "микроволновкой"

Снова предположим, что система RSA реализована на электронном устройстве (скажем, smart-карте), но теперь допустим, что секретное возведение в степень ($m \mapsto m^d \bmod n$ или $c \mapsto c^d \bmod n$) в процессе функционирования RSA использует китайскую теорему об остатках (теорема A.19); см. пример 9.1 (часть 4). Таким образом, мы считаем, что на этом устройстве выполняются два независимых возведения в степень — по модулю p и по модулю q, где $n = p \cdot q$.

Предположим теперь, что RSA используется для подписывания данных (это простейшая версия атаки; см. [LensA96] и [BoDML97]). Поэтому обычно атакующий представляет smart-карте сообщение m и ждет обратно $c = m^d \bmod n$. При этом, когда smart-карта выполняет свои вычисления, атакующий подвергает ее облучению надлежащего сорта ("просто положи ее в микроволновку" — сверхупрощение этой атаки) и надсется, что одно из двух возведений в степень будет сделано неправильно.

Например, smart-карта правильно вычисляет $c_1 = m^d \bmod p$, но находит неверное значение c_2, т.е. получает $c_2' \neq m^d \bmod q$. Читатель должен вспомнить, что в smart-карте записаны значения a и b, удовлетворяющие сравнениям

$$\begin{cases} a \equiv 1 & (\bmod p), \\ a \equiv 0 & (\bmod q); \end{cases}$$

$$\begin{cases} b \equiv 0 & (\bmod p), \\ b \equiv 1 & (\bmod q). \end{cases}$$

Таким образом, карта выдаст $c' = (a \cdot c_1 + b \cdot c_2') \bmod n$. Заметим теперь, что в силу сравнений $b \equiv 0 \bmod p$ и $a \equiv 0 \bmod q$

$$\begin{aligned} c - c' &\equiv a \cdot c_1 - a \cdot c_1 = 0 \quad (\bmod p), \\ c - c' &\equiv b \cdot c_2 - b \cdot c_2' = b(c_2 - c_2') \not\equiv 0 \quad (\bmod q). \end{aligned}$$

Отсюда следует, что НОД$(c - c', n)$ даст нетривиальное разложение n.

От способа применения карты зависит, сможет ли атакующий узнать также и правильное значение c, например, снова предлагая карте подписать m, но уже без всякого облучения. Способ обойти эту проблему состоит в том, что атакующий выбирает сообщение c, вычисляет $m = c^e \bmod n$

с публичным показателем e и предлагает это m при атаке на карту. В этом случае правильное значение c известно уже заранее.

Пример 9.1 (часть 6)

Продолжаем работу с параметрами примера 9.1, *так что* $p_B = 9733$, $q_B = 10177$, $n_B = 99052741$, $e_B = 81119923$, $d_B = 17089915$.

Далее, $a = 45287650$, $b = 53765092$ *(см. пример* 9.1 *(часть* 4*)). Возьмем* $c = 11111111$.

```
n = 99052741; e = 81119923;
c = 11111111;
m = PowerMod[c, e, m]
```

```
24307114
```

Таким образом, подписывая $m = 24307114$, *карта должна выдавать* $c = 11111111$. *Карта вычисляет числа* c_1 *и* c_2 *и получает с следующим образом:*

```
p = 9733; q = 10177;
d = 17089915; d1 = Mod[d, p - 1]; d2 = Mod[d, q - 1];
m1 = Mod[m, p]; m2 = Mod[m, q];
a = 45287650; b = 53765092;
c1 = PowerMod[m1, d1, p];
c2 = PowerMod[m2, d2, q];
c = Mod[a * c1 + b * c2, n]
```

```
11111111
```

Однако, если благодаря облучению c_1 *вычислено неправильно, скажем, получено* $c_1' = 8765$, *то карта будет продуцировать неправильное значение* c' *вместо* $c = 11111111$, *и* НОД *разности этих двух чисел с* n *даст делитель числа* n.

```
c1Prime = 8765;
cPr = Mod[a * c1Prime + b * c2, n]
GCD[c - cPr, n]
```

```
92608527
```

```
10177
```

Действительно, число 10177 — *один из двух делителей* n.

9.4 Как генерировать большие простые числа; некоторые тесты на простоту

9.4.1 Испытание случайных чисел

Чтобы практически построить систему RSA, нужен эффективный способ генерации очень длинных простых чисел. Следующий псевдоалгоритм описывает веротностный способ того, как это можно сделать.

Алгоритм 9.3 (*генерация l-значных простых чисел*)

Шаг 1: Написать случайное нечетное l-значное целое u.

Шаг 2: Проверить кандидата u на простоту; если u не простое, вернуться к шагу 1; иначе STOP.

В двух ближайших разделах мы обсудим некоторые способы проверки целого u на простоту. Первые два теста не дают абсолютной гарантии простоты u, но вероятность того, что составное число u пройдет тест, может быть сделана сколь угодно малой. Еще один тест (набросок которого дается в подразд. 9.4.3) может гарантировать простоту, но он намного медленнее. Другие тесты читатель может найти в разд. 4.5.4 книги [Knut81].

Пример 9.12

Чтобы смоделировать алг. 9.3 в пакете "Mathematica", можно использовать функции Random, PrimeQ и While. Заметим, что ниже четность u не проверяется (это в любом случае не является существенной частью алгоритма).

```
u = 1; l = 3; att = 0;
While[PrimeQ[u] == False, att = att + 1;
       u = Random[Integer, {10^(l-1), 10^l}]];
Print["простым числом является ", u]
Print[att, " попыток"]
```

```
простым числом является 907
7 попыток
```

Каково ожидаемое число пробегания шагов 1 и 2 в приведенном выше "алгоритме" до получения простого числа? Чтобы ответить на этот вопрос, мы должны знать долю простых чисел в множестве l-значных нечетных чисел. С этой целью напомним закон распределения простых чисел (теорема А.2).

Теорема 9.4
Пусть $\pi(x)$ подсчитывает число простых чисел, меньших или равных x (см. определение А.1). Тогда

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1.$$

С помощью закона распределения простых чисел (ЗРП) довольно легко получить долю нечетных l-значных чисел, которые просты. Вычисляем:

$$\frac{\pi(10^l) - \pi(10^{l-1})}{(10^l - 10^{l-1})/2} \overset{\text{ЗРП}}{\approx} \frac{10^l/\ln 10^l - 10^{l-1}/\ln 10^{l-1}}{(10^l - 10^{l-1})/2} = \frac{2(9\cdot l - 10)}{9\cdot l(l-1)\ln 10} \approx$$

$$\approx \frac{2}{l\cdot \ln 10}.$$

Например, при $l = 100$ получится

```
l = 100;
EstimateProb[l_] = 2*(9*l - 10)/(9*l*(l - 1)*Log[10]);
N[EstimateProb[100], 3]
```

```
0.00868
```

Так как обратное к этому числу приблизительно равно 115, ожидаемое число проходов в нашем "алгоритме" порождения простых чисел оценивается числом 115.

9.4.2 Вероятностные тесты простоты

□ Тест простоты Соловэя–Штрассена

Пусть p — простое число. Напомним (см. определение А.9), что целое число u, где $p \nmid u$ (читается: p не делит u), называется *квадратичным вычетом* (QR) по модулю p, если уравнение

$$x^2 \equiv u \pmod p$$

имеет целочисленное решение. Если $p \nmid u$ и данное сравнение не имеет целочисленного решения, то u называется *квадратичным невычетом* (NQR) по модулю p. Хорошо известный символ Лежандра $\left(\frac{u}{p}\right)$ (см. определение А.10) задается равенством

$$\left(\frac{u}{p}\right) = \begin{cases} +1, & \text{если } u \text{ — квадратичный вычет по модулю } p, \\ -1, & \text{если } u \text{ — квадратичный невычет по модулю } p, \\ 0, & \text{если } p \text{ делит } u. \end{cases}$$

Символ Якоби $\left(\frac{u}{m}\right)$ (см. определение А.11) обобщает символ Лежандра на все нечетные натуральные числа m. Пусть $m = \prod_i p_i^{e_i}$, где p_i (необязательно различные) нечетные простые числа. Тогда $\left(\frac{u}{m}\right)$ определяется равенством

$$\left(\frac{u}{m}\right) = \prod_i \left(\frac{u}{p_i}\right)^{e_i}.$$

В разд. А.4 читатель может найти все свойства символов Лежандра и Якоби. Эти свойства достигают кульминации в крайне эффективном алгоритме вычисления значений этих символов. Пример можно найти там же. В пакете "Mathematica" оба символа вычисляются с помощью функции *JacobiSymbol*:

```
u = 12703; m = 16361; JacobiSymbol[u, m]
```

```
|| 1
```

Так как m в примере выше — простое число, фактически легко вычислить квадратный корень из u. Читатель отсылается к разд. 9.5, где обсуждается, как именно это сделать. В пакете "Mathematica" можно просто использовать функцию *Solve*.

```
Clear[x];
Solve[{x^2 == 12703, Modulus == 16361}, x]
```

```
|| {Modulus→16361, x→7008}, {Modulus→16361, x→9353}
```

В самом деле, $(\pm 7008)^2 \equiv 12703 \pmod{16361}$, что можно проверить с помощью функции *PowerMod*.

```
PowerMod[7008, 2, 16361]
```

```
|| 12703
```

Нахождение решений сравнения $x^2 \equiv u \pmod{m}$ для составных чисел m является, вообще говоря, очень трудной задачей, неподдающейся решению для больших значений m (обсуждение этой проблемы см. в [Pera86]).

Если m — произведение различных простых чисел, и это разложение известно (!), то можно найти квадратный корень из u по модулю m, найдя квадратные корни из u по модулю всех простых делителей числа m, а затем скомбинировав результаты посредством китайской теоремы об остатках. Этот метод будет продемонстрирован в разд. 9.5. Когда разложение m содержит более высокие степени простых чисел, все становится намного более сложным.

Пусть p — простое число, $p > 2$. Напомним, что по теореме А.23 для всех целых чисел u

$$\left(\frac{u}{p}\right) \equiv u^{(p-1)/2} \pmod{p}. \tag{9.15}$$

Алгоритм Соловэя–Штрассена [SolS77] опирается на следующую теорему:

Теорема 9.5
Пусть m — нечетное целое число, а G определяется равенством

$$G = \left\{0 < u < m \,\middle|\, \text{НОД}(u,m) = 1 \text{ и } \left(\frac{u}{m}\right) \equiv u^{(m-1)/2} \pmod{m}\right\}.$$

Тогда

$$|G| = m - 1, \quad \text{если} \quad m \quad \text{просто}; \tag{9.16}$$

$$|G| \leq (m-1)/2, \quad \text{если} \quad m \quad \text{не просто}. \tag{9.17}$$

Доказательство. Если m просто, то любое целое $u, 0 < u < m$, удовлетворяет сравнению (9.15) и имеет НОД с m, равный 1, так что в этом случае $|G| = m - 1$.

Поэтому нужно рассмотреть лишь случай, когда m не просто. Очевидно, G — подгруппа мультипликативной группы

$$\mathbb{Z}_m^* = \{0 < u < m | \text{НОД}(u,m) = 1\}.$$

Из теоремы В.5 следует, что мощность G делит мощность $\mathbb{Z}_m^*$. Поэтому, если $G \neq \mathbb{Z}_m^*$, можно заключить, что

$$|G| \leq |\mathbb{Z}_m^*|/2 = \varphi(m)/2 \leq (m-1)/2.$$

Это докажет теорему. Таким образом, достаточно доказать существование такого элемента u в $\mathbb{Z}_m^*$, что $\left(\frac{u}{m}\right) \not\equiv u^{(m-1)/2} \pmod{m}$.

Мы различаем два случая. В [SolS77] авторы опускают рассмотрение случая, когда m — квадрат. В приводимом ниже доказательстве, принадлежащем Дж.У.Найенхьюзу (частное сообщение), случай 1 покрывает опущенную возможность.

Случай 1: число m делится на квадрат некоторого простого числа.

Запишем $m = p^r \cdot s$, где p — нечетное простое, $r \geq 2$ и $\text{НОД}(p,s) = 1$. Пусть u — решение системы сравнений

$$u \equiv 1 + p \pmod{p^r}, \tag{9.18}$$

$$u \equiv 1 \pmod{s}. \tag{9.19}$$

В силу китайской теоремы об остатках (теорема А.19) такое решение u существует и единственно по модулю m. Очевидно, $\text{НОД}(u, p^r) =$

$=$ НОД$(u, s) = 1$ и, значит, НОД$(u, m) = 1$, т.е. $u \in \mathbb{Z}_m^*$. Из сравнения (9.18) и биномиальной теоремы с помощью рассуждения, подобного доказательству теоремы B.26, следует, что $u^m \equiv (1+p)^m \equiv 1 \pmod{p^r}$. В силу соотношения (9.19) мы также имеем $u^m \equiv 1 \pmod{s}$. Теперь китайская теорема об остатках дает $u^m \equiv 1 \pmod{m}$.

Так как ввиду (9.18) $u \not\equiv 1 \pmod{m}$, из сказанного вытекает, что $u^{m-1} \not\equiv 1 \pmod{m}$. Это, в свою очередь, влечет $u^{(m-1)/2} \not\equiv \pm 1 \pmod{m}$, откуда следует, что u не может удовлетворять сравнению (9.15). Мы заключаем, что элемент u лежит в $\mathbb{Z}_m^*$, но не в G.

Случай 2: m — произведение s различных простых чисел, скажем, $m = p_1 p_2 \ldots p_s$, где $s \geq 2$.

Пусть a — квадратичный невычет по модулю p_1. В силу китайской теоремы об остатках существует целое число u, единственное по модулю m, удовлетворяющее системе сравнений

$$u \equiv a \pmod{p_1}, \tag{9.20}$$

$$u \equiv 1 \pmod{p_i}, \quad 2 \leq i \leq s. \tag{9.21}$$

Очевидно, НОД$(u, p_i) = 1, 1 \leq i \leq s$, так что $u \in \mathbb{Z}_m^*$. Чтобы убедиться в том, что $u \notin G$, нужно показать, что для этого u соотношение (9.15) не выполняется.

Из сравнений $u \equiv 1 \pmod{p_i}$, $2 \leq i \leq s$, следует, что для этих индексов $\left(\frac{u}{p_i}\right) = 1$. Но $\left(\frac{u}{p_1}\right) = \left(\frac{a}{p_1}\right) = -1$, так как a — квадратичный невычет. Из определени символа Якоби (определение A.11) вытекает, что $(u/m) = -1$. Это, в частности, влечет, что $\left(\frac{u}{m}\right) \equiv -1 \pmod{p_i}$, когда $2 \leq i \leq s$.

С другой стороны, (9.21) показывает, что $u^{(m-1)/2} \equiv 1 \pmod{p_i}, 2 \leq \leq i \leq s$. Следовательно,

$$\left(\frac{u}{m}\right) \not\equiv u^{(m-1)/2} \pmod{p_i}$$

для всех $i, 2 \leq i \leq s$, и тем более не выполняется сравнение (9.15).

∎

Теперь можно описать алгоритм Соловэя–Штрассена.

Алгоритм 9.6 (*тест простоты Соловэя–Штрассена*)

input нечетное целое число m (кандидат),
параметр безопасности k;

initialize prime = True, $j = 1$;

while prime **and** $i \le k$ **do**
begin
select случайное целое u;
if НОД$(u, m) \neq 1$ **or** $\left(\frac{u}{m}\right) \not\equiv u^{(m-1)/2} \pmod{m}$
then prime = False;
$i = i + 1$;
end
output prime.

В этом алгоритме k — произвольное натуральное число. В силу теоремы 9.5 вероятность того, что k независимо и случайно выбранных элементов u проходят тест, заданный алг. 9.6, когда m — составное, не превосходит 2^{-k}. При достаточно большом k вероятность того, что составное число "выдерживает" приведенный алгоритм, может быть сделана сколь угодно малой.

См., однако, в следующем пункте алгоритм Миллера–Рабина, где вероятность того, что составность числа не обнаруживается после k тестирований, равна 4^{-k}.

Пример 9.13

Чтобы проверить, является ли нечетное число $m = 1234563$ *простым, можно использовать функции GCD, JacobiSymbol, PowerMod и Mod из пакета "Mathematica":*

```
m = 1234563; u = 1212121; GCD[u, m] == 1
Mod[JacobiSymbol[u, m] - PowerMod[u, (m - 1)/2, m], m] == 0
```

```
True
```

```
False
```

Предлагаем читателю проверить на простоту число $m = 104729$.

□ Тест Миллера–Рабина

Тест Миллера–Рабина [Mill76], [Rabi80a] основан на том факте (см. теорему B.15), что при простом p уравнение $x^2 \equiv 1 \pmod{p}$ имеет лишь два решения: $x \equiv \pm 1 \pmod{p}$.

Итак, пусть m — нечетное целое число, которое мы хотим тестировать на простоту. Допустив на миг, что m — действительно простое, по теореме Ферма (теорема A.15) получим, что любое целое a с НОД$(a, m) = 1$ удовлетворяет сравнению $a^{m-1} \equiv 1 \pmod{m}$.

Так как $m - 1$ четно, получаем $a^{(m-1)/2} \equiv \pm 1 \pmod{m}$. Если оказывается, что $a^{(m-1)/2}$ сравнимо с $+1$ и $(m-1)/2$ четно, то можно повторить рассуждение, заключив в этом случае, что $a^{(m-1)/4} \equiv \pm 1 \pmod{m}$ и т.д. На этом пути можно доказать следующую лемму.

Лемма 9.7
Взяв простое p, запишем $p-1 = a \cdot 2^f$, где a нечетно. Пусть u — целое число между 1 и $p-1$. Тогда либо $u^a \equiv 1 \pmod{p}$, либо $u^{a \cdot 2^i} \equiv -1 \pmod{p}$ для некоторого $i, 0 \leq i < f$.

Чтобы проверить на простоту нечетное целое число m, поступим следующим образом. Сначала запишем $m-1 = a \cdot 2^f$, где a нечетно. Далее выберем случайное целое число $u, 2 \leq u < m$, и вычислим $u^a, u^{a \cdot 2}, \ldots, u^{a \cdot 2^f}$. Как только одно из этих чисел не попадает в $\{-1, 1\}$, хотя следующее равно $+1$, или же $u^{a \cdot 2^f} \not\equiv 1 \pmod{m}$, мы заключаем, что m — составное и можно остановиться. Повторяем этот тест k раз, где k — параметр безопасности, который мы вскоре обсудим.

Пусть m — целое число, а u таково, что $u^{a \cdot 2^j} \equiv 1 \pmod{m}$, тогда как $u^{a \cdot 2^{j-1}} \not\equiv \pm 1 \pmod{m}$. Тогда u называется *сильным свидетелем* составности m. Оно доказывает, что m — составное. С другой стороны, пусть m — составное, а u — целое число, удовлетворяющее сравнению $u^a \equiv 1 \pmod{m}$ или сравнению $u^{a \cdot 2^j} \equiv -1 \pmod{m}$ для некоторого $j, 0 \leq j < f$; тогда u называется *сильным лгуном* (относительно простоты) числа m[7].

Для эффективности теста простоты желательно, чтобы составные числа имели как можно меньше сильных лгунов.

Алгоритм 9.8 (*тест простоты Миллера–Рабина*)

input нечетное целое число m (кандидат),
параметр безопасности k;
initialize prime = True, $i = 1$;
write $m - 1 = a \cdot 2^f$, где a нечетно;
while prime **and** $i \leq k$ **do**
begin
select случайное целое $u, 1 < u < m-1$;
compute $x = u^a \bmod m$;
if $x \not\equiv \pm 1 \pmod{m}$ **then**
begin **put** $j = 1$;
while $x \not\equiv \pm 1 \pmod{m}$ **and** $j \leq f-1$ **do**
begin $x \leftarrow x^2 \bmod m$;
if $x \equiv 1 \pmod{m}$ **then** prime = False;
$j \leftarrow j + 1$
end
$i = i + 1$
end
output prime.

[7] В последнем случае число m называют *сильным* (или *сильно*) *псевдопростым* по основанию u.— *Прим. ред.*

Пример 9.14

Пусть $m = 7933$. *Тогда* $m - 1 = 1983 \cdot 2^2$. *Выбираем случайное* u *и вычисляем* $u^{1983 \cdot 2^i}$ *при* $i = 0, 1, 2$. *Для записи* $m - 1$ *в виде* $a \cdot 2^f$ *используются функции* While *и* EvenQ *из пакета "Mathematica", а для тестирования — функции* Random, PowereMod, Print, Mod *и* Do.

```
m = 7933;
f = 0; a = m-1; While[EvenQ[a], f = f+1, a=a/2];
{a, f}
u = Random[Integer, {1, m - 2}]
x = PowerMod[u, a, m];
Do[{Print[x], x = Mod[x^2, m]}, {i, 0, f}]
```

```
{1983, 2}

4225

7932        1        1
```

Видно, что как бы часто мы ни пробегали алгоритм, мы всегда получим $(+1, +1, +1)$, *или* $(-1, +1, +1)$, *или* $(*, -1, +1)$.

Пример 9.15

Пусть $m = 429$. *Как мы увидим ниже, выбор* $u = 34$ *дает сильного свидетеля составности* m.

```
m = 429;
f = 0; a = m - 1;
While[EvenQ[a], f = f + 1, a = a / 2];
{a, f}
u = 34;
x = PowerMod[u, a, m];
Do[{Print[x], x = Mod[x^2, m]}, {i, 0, f}]
```

```
{107, 2}

265        298        1
```

Что остается сделать, так это дать оценку доли сильных лгунов по модулю составного числа. Следующая теорема утверждает, что эта доля составляет самое большее $(1/4)$. Это означает, что вероятность того, что составность числа не будет обнаружена после k проходов теста Миллера–Рабина, не превосходит $(1/4)^k$. Это гораздо лучше аналогичной вероятности для теста простоты Соловэя–Штрассена, которая ограничена сверху лишь числом $(1/2)^k$.

Теорема 9.9
Пусть m — составное число, $m \neq 9$. Тогда число сильных лгунов, лежащих между 1 и $m-1$, не превосходит $\varphi(m)/4$, где φ — функция Эйлера. Иными словами, вероятность того, что после k проходов алг. 9.8 не обнаруживает составность составного числа m, не превосходит 4^{-k}.

Доказательство теоремы 9.9 (см. [Moni80] или [Rabi80a]) очень технично и не добавляет понимания читателю этого введения[8].

Если $m = 9$, то $\varphi(m)/4$ равно $6/4$, что меньше 2. Однако имеются именно 2 сильных лгуна: $+1$ и -1.

9.4.3 Детерминированный тест простоты

Тесты простоты, которые детерминированным способом доказывают простоту данного числа, конечно, много медленнее, чем вероятностные алгоритмы типа тех, что обсуждались в предыдущем подразделе.

Здесь мы объясним идею, лежащую в основе детерминированного теста простоты Коэна и Ленстры мл. [CohL82]. Этот тест является улучшением теста из [AdPR83]. Мы не даем полного описания теста. Это потребовало бы слишком много продвинутой и глубокой теории чисел[9]. Мы близко следуем блестящей вводной статье Ленстры [LensH83].

Начнем с напоминания теоремы Ферма (теорема А.15).

Теорема 9.10 (*Ферма*)
Пусть m — простое число, a — произвольное целое число. Тогда

$$a^m \equiv a \pmod{m}. \tag{9.22}$$

Пусть m — натуральное число и мы хотим проверить его на простоту. Всего лишь одно целое a, не удовлетворяющее сравнению (9.22), опровергает простоту m.

К сожалению, обратное не верно. Например, число $m = 561$ удовлетворяет соотношению (9.22), хотя $m = 3 \times 11 \times 17$. Чтобы увидеть это, вычислим сначала $\text{НОК}[\varphi(3), \varphi(11), \varphi(17)] \overset{A.17}{=} \text{НОК}[2, 10, 16] = 80$. Пусть a взаимно просто с 561. Из теоремы Эйлера (теорема А.14) следует, что a^{80} сравнимо с единицей по модулю каждого из трех простых делителей числа 561. Теперь китайская теорема об остатках (теорема А.19) влечет, что $a^{80} \equiv 1 \pmod{561}$. Следовательно, $a^{561} = a \cdot (a^{80})^7 \equiv a \pmod{561}$.

Для значений a, имеющих общий делитель с числом 561, сравнение (9.22) доказывается аналогичным образом.

---

[8] Тем не менее, мы рекомендуем читателю ознакомиться с подробным доказательством на русском языке, которое можно найти в *[НодК99], разд. 6.2.— *Прим. ред.*

[9] Конкретнее, требуется хорошее знание свойств колец p-адических чисел и полей деления круга.— *Прим. ред.*

Читатель может проверить вышесказанное с помощью функций *FactorInteger* и *PowerMod* из пакета "Mathematica":

```
m = 561; FactorInteger[m]
a = 543; PowerMod[a, m, m] == a
```

```
{{3, 1}, {11, 1}, {17, 1}}
```

```
True
```

Составные числа m, обладающие тем свойством, что $a^{m-1} \equiv 1 \pmod{m}$ для всех a с $\text{НОД}(a, m) = 1$, называются *числами Кармайкла*.

Однако для несколько более сильного утверждения, чем теорема 9.10, обратное выполняется. В последующем $\left(\frac{a}{m}\right)$ обозначает, как обычно, символ Якоби.

Теорема 9.11
Нечетное число m просто тогда и только тогда, когда для всех целых чисел a

$$\text{НОД}(a, m) = 1 \Longrightarrow a^{(m-1)/2} \equiv \left(\frac{a}{m}\right) \pmod{m}.$$

Доказательство. То, что указанное соотношение выполняется для простых чисел, было уже отмечено в соотношении (9.15). Обратное было первоначально доказано Лемером [Lehm76], но оно также прямо следует из теоремы 9.5.

■

Последняя теорема, конечно, не является эффективным критерием простоты для чисел длины более 100 цифр. Ленстра предложил следующую "заманчивую" альтернативу.

Теорема 9.12
Нечетное число m просто тогда и только тогда, когда любой делитель d числа m является степенью m.

Доказательство. Это утверждение вполне тривиально, так как $d = 1 = m^0$ и $d = m = m^1$, и это — единственные (натуральные) делители простого числа m. Все другие числа между 1 и m не могут быть записаны как степени m.

■

Ясно, что это не та теорема, которую мы хотели бы использовать в качестве критерия простоты, но одна из ее вариаций оказывается очень полезной. Мы покажем, что при некоторых условиях любой делитель числа m немного походит на степень m.

Теорема 9.13
Пусть целое число m взаимно просто с 6. Далее, допустим, что

$$\left(\frac{u}{m}\right) \equiv u^{(m-1)/2} \pmod{m} \quad \text{для } u = -1, 2, 3; \tag{9.23}$$

$$a^{(m-1)/2} \equiv -1 \pmod{m} \quad \text{для некоторого целого } a. \tag{9.24}$$

Тогда для каждого d, делящего m,

$$d \equiv m^j \pmod{24} \quad \text{для некоторого неотрицательного } j. \tag{9.25}$$

Фактически соотношение (9.25) можно усилить до сравнения

$$d \equiv m^j \pmod{24} \quad \text{для } j = 0 \text{ или } j = 1. \tag{9.26}.$$

Условие (9.24) в этой теореме не может быть опущено. В самом деле, число $m = 1729 = 7 \times 13 \times 19$ удовлетворяет (9.23), но не удовлетворяет (9.25). Заметим, что $m \equiv 1 \pmod{24}$; поэтому никакая степень m не может быть равна какому-либо простому делителю числа m.

Все эти утверждения можно проверить с помощью функций *FactorInteger*, *JacobiSymbol*, *PowerMod* и *Mod* пакета "Mathematica":

```
m = 1729; FactorInteger[m]
Mod[m, 24]
Mod[JacobiSymbol[-1,m] - PowerMod[-1,(m-1)/2,m],m] == 0
Mod[JacobiSymbol[2, m] - PowerMod[2,(m-1)/2,m],m] == 0
Mod[JacobiSymbol[3, m] - PowerMod[3,(m-1)/2,m],m] == 0
```

```
|| {{7, 1}, {13, 1}, {19, 1}}
|| 1
|| True
|| True
|| True
```

Прежде, чем доказывать теорему 9.13, проиллюстрируем, как ее можно использовать для тестирования на простоту целых чисел m, $24 < m < 24^2$. После доказательства мы обсудим обобщения теоремы 9.13, которые дают эффективные тесты простоты для бóльших значений m.

Алгоритм 9.14 (*тест простоты Коэна–Ленстры*)

input $m, 24 < m < 24^2$;
initialize prime = True;

test 1: **if** НОД$(m, 6) \neq 1$ **then** prime = False;
test 2: **if** $\left(\frac{u}{m}\right) \not\equiv u^{(m-1)/2} \pmod{m}$ для $u = -1, 2, 3$ **then** prime = False;
test 3: найти такое целое a, что $a^{(m-1)/2} \equiv -1 \pmod{m}$; **if** такого a не существует **then** prime = False;
test 4: вычислить $d = m \bmod 24$; **if** $d > 1$ **and** $d|m$ **then** prime = False;
output prime.

Доказательство. Первым делом обратимся к тесту 3. Если m просто, то вероятность того, что случайное $a, 1 < a < m$, удовлетворяет условию (9.24), равна 1/2 в силу теорем А.23 и А.20. Таким образом, можно ожидать, что в среднем нужны две попытки для нахождения целого числа a, удовлетворяющего соотношению (9.24). Если такое целое a не существует, то m не просто.

Об этом шаге можно сказать больше. Предполагая истинной расширенную гипотезу Римана, можно даже доказать, что если m просто, то сравнение (9.24) имеет такое решение a, что $1 < a < 2(\log m)^2$. (См. также [Pera86].)

Если m проходит первые три теста, то, согласно теореме 9.13, каждый делитель d числа m должен быть сравним с 1 или с m по модулю 24. Так как $m < 24^2$, можно предположить, что $d < 24$ (в противном случае вместо d рассмотрим m/d). Отсюда следует, что фактически d *равно* 1 или $m \bmod 24$. Возможность равенства $d = m \bmod 24$ при $d > 1$ исключается тестом 4. Отсюда следует, что делитель d должен быть равен 1. Мы заключаем, что m просто.

■

Для доказательства теоремы 9.13 нам нужны следующие две леммы. Первая дает необходимое и достаточное условие для того, чтобы два целых числа m_1 и m_2, взаимно простых с 6, были сравнимы по модулю 24.

Лемма 9.15
Пусть m_1 и m_2 — натуральные числа, взаимно простые с 6. Тогда

$$m_1 \equiv m_2 \pmod{24} \iff \left(\frac{u}{m_1}\right) = \left(\frac{u}{m_2}\right) \quad \text{для } u = -1, 2, 3.$$

Доказательство. Существуют восемь чисел $m, 1 \leq m \leq 24$, взаимно простых с 24, а именно, 1, 5, 7, 11, 13, 17, 19 и 23. Для каждого из этих m мы вычисляем значения $\left(\frac{u}{m}\right)$ при $u = -1, 2, 3$ с помощью следствия А.24, теоремы А.25 и теоремы А.27, или же в пакете "Mathematica" с помощью функции *JacobiSymbol*, которую можно применить сразу ко всему списку чисел.

```
m = {1, 5, 7, 11, 13, 17, 19, 23};
JacobiSymbol[-1, m]
JacobiSymbol[2, m]
JacobiSymbol[3, m]
```

```
|| {1, 1, -1, -1, 1, 1, -1, -1}
|| {1, -1, 1, -1, -1, 1, -1, 1}
|| {1, -1, -1, 1, 1, -1, -1, 1}
```

Легко проверить, что матрица с этими тремя векторами-строками обладает тем свойством, что все ее столбцы различны. Это показывает, что три значения $\left(\frac{u}{m}\right), u = -1, 2, 3$, однозначно определяют m в множестве $\{1, 5, 7, 11, 13, 17, 19, 23\}$.

∎

Например, глядя на второй столбец, мы видим, что $m = 5$ однозначно определяется в $\{1, 5, 7, 11, 13, 17, 19, 23\}$ тремя значениями $\left(\frac{-1}{m}\right) = 1$, $\left(\frac{2}{m}\right) = -1$ и $\left(\frac{2}{m}\right) = -1$.

Лемма 9.16
Пусть m — произвольное целое число. Тогда

$$\text{НОД}(m, 6) = 1 \Longrightarrow m^2 \equiv 1 \pmod{24}.$$

Доказательство. Так как m не делится на 3, получаем $m^2 \equiv 1 \pmod 3$. Аналогично, из нечетности m следует, что $m^2 \equiv 1 \pmod 8$. Чтобы увидеть это, запишем $m = 2n + 1$. Тогда $m^2 = (2n + 1)^2 = 4n(n + 1) + 1$.

Поскольку 3 и 8 взаимно просты, наше утверждение следует из китайской теоремы об остатках.

∎

Конечно, последнюю лемму можно проверить с помощью функции *Mod* из пакета "Mathematica" следующим образом:

```
m = {1, 5, 7, 11, 13, 17, 19, 23};
Mod[m^2, 24]
```

```
|| {1, 1, 1, 1, 1, 1, 1, 1}
```

Теперь мы готовы доказать теорему 9.13.

Доказательство теоремы 9.13. То, что показатель j в (9.25) может быть приведен по модулю 2, прямо следует из условия $\text{НОД}(m, 6) = 1$ и леммы 9.16. Это показывает, что сравнение (9.25) можно заменить на (9.26).

Далее, заметим, что (9.25) достаточно доказать только для простых делителей d числа m. Запишем $m - 1 = f \cdot 2^k$ и $d - 1 = g \cdot 2^\ell$, где f и g нечетны и $k, \ell > 0$.

Докажем сначала, что $\ell \geq k$, а затем используем лемму 9.15, чтобы показать, что либо $d = m^0 \bmod 24$, либо $d = m^1 \bmod 24$.

Возведем обе части сравнения (9.24) в степень g и приведем результат по модулю d. Так как $d|m$ и g нечетно, получим

$$a^{f \cdot g \cdot 2^{k-1}} \equiv (-1)^g \equiv -1 \pmod{d}.$$

Поскольку мы предполагаем, что d просто и a не имеет общих делителей с d или с m, из теоремы Ферма (теорема А.15) следует, что

$$a^{f \cdot g \cdot 2^{\ell}} \equiv a^{f(d-1)} \equiv 1^f = 1 \pmod{d}.$$

Из двух последних сравнений вытекает, что $k - 1 < \ell$.

Рассмотрим теперь $u \in \{-1, 2, 3\}$. Так как g нечетно и $d|m$, имеем

$$u^{f \cdot g \cdot 2^{k-1}} \equiv u^{g(m-1)/2} \overset{(9.23)}{\equiv} \left(\frac{u}{m}\right)^g \equiv \left(\frac{u}{m}\right) \pmod{d}.$$

С другой стороны (снова ввиду простоты d),

$$u^{f \cdot g \cdot 2^{\ell-1}} \equiv u^{f(d-1)/2} \overset{(9.15)}{\equiv} \left(\frac{u}{p}\right)^f \equiv \left(\frac{u}{p}\right) \pmod{d}.$$

Из двух последних сравнений вытекает, что для $i = -1, 2, 3$

$$\left(\frac{u}{p}\right) = \left(\frac{u}{m}\right)^{2^{\ell-k}}. \tag{9.27}$$

Заметим, что мы заменили отношение сравнения равенством. Это можно сделать, потому что обе части принимают значения ± 1.

Если $\ell = k$, то равенство (9.15) и лемма 9.15 влекут за собой, что $d \equiv m = m^1 \pmod{24}$. С другой стороны, если $\ell > k$, то правая часть в (9.27) равна 1, что также есть $(u/1)$. Поэтому лемма 9.15 дает $d \equiv 1 = m^0 \pmod{24}$.

■

В применении теоремы 9.13 решающим является тот факт, что условие (9.25) можно заменить на (9.26). Ввиду этого в четырех шагах алг. 9.14 только одно условие нуждается в проверке. Основанием для замены (9.25) на (9.26) служит то (см. лемму 9.16), что

$$\text{НОД}(m, 24) = 1 \Longrightarrow m^2 \equiv 1 \pmod{24}.$$

Теорема 9.13 проверяет простоту только таких чисел m, что $24 < m < 24^2$. Для бóльших значений m необходимы обобщения теоремы 9.13. Как можно ожидать, в этих обобщениях показатель в лемме 9.16

должен возрастать. Примером такого обобщения может служить импликация

$$\text{НОД}(m, 65520) = 1 \Longrightarrow m^{12} \equiv 1 \pmod{65520}.$$

Для проверки на простоту 100-значного числа m используется импликация

$$\text{НОД}(m, s) = 1 \Longrightarrow m^{5040} \equiv 1 \pmod{s},$$

где s — 53-значное число

$$2^6 \times 3^3 \times 5^2 \times 7^2 \times 11 \times 13 \times 17 \times 19 \times 31 \times 37 \times 41 \times 43 \times 61 \times 71 \times 73\times$$

$$\times 113 \times 127 \times 181 \times 211 \times 241 \times 281 \times 337 \times 421 \times 631 \times 1009 \times 2521.$$

Заметим, что $\sqrt{m} < s$, если m имеет не более 100 цифр. Приведем грубый набросок теста простоты для 100-значного числа[10].

Алгоритм 9.17 (*набросок теста простоты Коэна–Ленстры*)

input натуральное $m < 10^{100}$;
initialize prime = True;
test 1: **if** НОД$(m, s) \neq 1$ **then** prime = False;
test 2: **if** m не удовлетворяет одному из 67 сравнений, подобных (9.23) **then** prime = False;
test 3: вычислить $d = m^i \bmod s$ для $i = 1, 2, \ldots, 5039$; **if** одно из этих d делит m, **then** prime = False;
output prime.

Если m — составное, то приведенный выше алгоритм иногда даст делитель числа m. Но вероятность такого события очень мала: в большинстве случаев составного m алгоритм будет заканчиваться на шаге 2, не "выдавая" делителей m. Этот алгоритм можно адаптировать для тестирования на простоту и бóльших целых чисел. Ожидаемым временем пробега служит

$$(\ln m)^{c \ln\ln\ln m},$$

где c — некоторая константа[11].

[10] Это *слишком* грубый набросок. Наверное, читателю лучше обратиться к русскому переводу первоисточника [CohL82].— *Прим. ред.*

[11] В августе 2002 г. в Internet появился препринт *[AgKS02] трех индийских математиков, где излагается элегантный детерминированный тест простоты с ожидаемым временем работы $O((\ln m)^{12} F(\ln\ln m))$, где F — некоторый многочлен. При этом верится, что показатель 12 может быть уменьшен до 6. Изложение этого теста на русском языке есть в книгах *[Чер02] и *[Вас03]).— *Прим. ред.*

9.5 Вариант Рабина

В подразд. 9.2.1 было упомянуто, что неизвестен никакой общий метод взлома RSA, кроме факторизации n. В [Rabi79] Рабин предложил вариант системы RSA, для которого можно доказать, что его криптоанализ эквивалентен факторизации модуля n.

9.5.1 Функция шифрования

В системе RSA каждый пользователь U должен выбрать публичный показатель e_U с НОД$(e_U, \varphi(n_U)) = 1$ (см. 9.1.2). В схеме Рабина все пользователи U имеют один и тот же показатель

$$e_U = 2. \tag{9.28}$$

Напомним читателю обсуждение в подразд. 9.3.1.

Так как НОД$(2, \varphi(n_U)) = 2$, и поскольку числа $p_U - 1$ и $q_U - 1$ четны, шифрование более не является взаимно однозначным отображением. В самом деле, если $c \equiv m^2 \pmod{n}$, где НОД$(c, n_U) = 1$ и $n_U = p_U q_U$, то сравнение $x^2 \equiv c \pmod{p_U}$ имеет два решения, а именно $\pm m \bmod p_U$ и, аналогично, сравнение $x^2 \equiv c \pmod{q_U}$ имеет два решения $\pm m \bmod q_U$. По китайской теореме об остатках (теорема А.19) сравнение

$$x^2 \equiv c \pmod{n_U} \tag{9.29}$$

имеет четыре решения по модулю n_U. Что происходит, когда НОД$(c, n_U) \neq \neq 1$ — легкое упражнение для читателя (см. задачу 9.12).

Пример 9.16 (часть 1)

Рассмотрим шифрование сообщения $m = 12345678$ *по модулю* $n = 9733 \times 10177 = 99052741$ *(мы используем функции* Prime *и* PowerMod *из пакета "Mathematica").*

```
pB = Prime[1200];
qB = Prime[1250];
nB = pB * qB
m = 12345678;
PowerMod[m, 2, nB]
```

```
99052741
```

```
43962531
```

Чтобы найти четыре сообщения, которые отображаются в один и тот же шифртекст, нужно с помощью китайской теоремы об остатках скомбинировать четыре системы линейных сравнений $x \equiv \pm m \pmod{p}$ *и* $x \equiv \pm m \pmod{q}$. *В "Mathematica" необходимо загрузить пакет NumberTheory'NumberTheoryFunction', чтобы воспользоваться функцией* ChinesRemainderTheorem.

```
<< NumberTheory`NumberTheoryFunctions`
```

```
m1 = ChinesRemainderTheorem[
    {12345678, 12345678}, {9733, 10177}]
m2 = ChinesRemainderTheorem[
    {-12345678, 12345678}, {9733, 10177}]
m3 = ChinesRemainderTheorem[
    {12345678, -12345678}, {9733, 10177}]
m4 = ChinesRemainderTheorem[
    {-12345678, -12345678}, {9733, 10177}]
```

```
|| 12345678

|| 48738630

|| 50314111

|| 86707063
```

Для проверки вычисляем

```
PowerMod[m1, 2, nB]
PowerMod[m2, 2, nB]
PowerMod[m3, 2, nB]
PowerMod[m4, 2, nB]
```

```
|| 43962531
```

и еще три точно таких же ответа.

Заметим, что пространство-образ функции шифрования не совпадает со всем множеством $\{0, 1, \ldots, n_U\}$. Вследствие этого вариант Рабина нельзя непосредственно использовать в схеме подписи (см. связанный с ним протокол Фиата–Шамира в гл. 14).

9.5.2 Дешифрование

□ Предвычисление

Как дешифровать сообщение $c \equiv m^2 \pmod{n}$ в варианте Рабина системы RSA? Раньше в данном разделе объяснялось, что мы делаем это с помощью китайской теоремы об остатках. В предвычислении определяются целые числа a и b, удовлетворяющие сравнениям

$$a \equiv 1 \pmod{p_U} \quad \text{и} \quad a \equiv 0 \pmod{q_U}, \tag{9.30}$$

$$b \equiv 0 \pmod{p_U} \quad \text{и} \quad b \equiv 1 \pmod{q_U}. \tag{9.31}$$

Решения a и b могут быть легко найдены следующим образом. Например, чтобы найти a, из второго сравнения получаем $a = l \cdot q_U$ и подставляем в

первое сравнение. Это приводит к сравнению $l \cdot q_U \equiv 1 \pmod{p_U}$, которое можно решить с помощью расширенной версии алгоритма Эвклида (алг. А.8). См. также пример А.3.

Такие системы сравнений можно непосредственно решать с помощью функции *ChinesRemainderTheorem* из пакета "Mathematica", для чего сначала нужно загрузить пакет *NumberTheory'NumberTheoryFunctions'*.

Пример 9.16 (часть 2)

Продолжая работу с параметрами примера 9.16, *мы должны решить системы сравнений*

$$\begin{array}{rcll} a & \equiv & 1 & \pmod{9733} \\ b & \equiv & 0 & \pmod{9733} \end{array} \quad u \quad \begin{array}{rcll} a & \equiv & 0 & \pmod{10177}, \\ b & \equiv & 1 & \pmod{10177}. \end{array}$$

```
<< NumberTheory`NumberTheoryFunctions`
```

```
a = ChinesRemainderTheorem[{1, 0}, {9733, 10177}]
b = ChinesRemainderTheorem[{0, 1}, {9733, 10177}]
```

```
45287650
```

```
53765092
```

Таким образом, $a = 45287650, b = 53765092$.

□ Нахождение квадратных корней по модулю простого числа

Нужно решить сравнение $x^2 \equiv c \pmod{p_U}$ (и, аналогично, $x^2 \equiv c \pmod{q_U}$). Если $c = 0$, то решение очевидно. Поэтому допустим, что $c \not\equiv 0 \pmod{p_U}$.

Начиная с этого момента, мы упрощаем обозначения, опуская индекс U. Оказывается, непосредственная техника вычисления x не всегда возможна. Мы рассмотрим три случая.

Случай 1: $p \equiv 3 \pmod{4}$.

Если c — квадрат некоторого элемента m из $\mathbb{Z}_p$ (такое c называется квадратичным вычетом по модулю p; см. разд. А.4), то два решения сравнения $x^2 \equiv c \pmod{p}$ даются выражением $\pm c^{(p+1)/4}$. Действительно, возводя это выражение в квадрат, по теореме Ферма получим

$$\left(\pm c^{(p+1)/4}\right)^2 \equiv c^{(p+1)/2} \equiv c \cdot c^{(p-1)/2} \equiv c \cdot m^{p-1} \overset{A.15}{\equiv} c \pmod{p}.$$

Пример 9.17

Рассмотрим простое число $p = 3571$, *которое сравнимо с* 3 *по модулю* 4. *Число* $c = 2868$ *является кадратичным вычетом по модулю* p, *что можно проверить с помощью символа Лежандра. Для проверки всех*

этих утверждений используем функции Prime, Mod и JacobiSymbol из пакета "Mathematica".

```
p = Prime[500]
Mod[p, 4] == 3
c = 2868;
JacobiSymbol[c, p] == 1
```

```
3571
```

```
True
```

```
True
```

Решение сравнения $x^2 \equiv 2868 \pmod{p}$ *дается формулами* $m \equiv \pm 2868^{(p+1)/4} \equiv \pm 3234 \pmod{3571}$. *Чтобы убедиться в этом, используем функцию PowerMod.*

```
m = PowerMod[c, (p + 1) / 4, p]
PowerMod[{m, -m}, 2, p]
```

```
3234
```

```
{2868}, {2868}
```

Случай 2: $p \equiv 5 \pmod{8}$.

С помощью немного уточненного метода, использованного выше, можно показать, что в данном случае решение сравнения $x^2 \equiv c \pmod{p}$ дается формулой $\pm c^{(p+3)/8}$, если $c^{(p-1)/4} \equiv 1 \pmod{p}$, и формулой $\pm 2c \cdot (4c)^{(p-5)/8}$, если $c^{(p-1)/4} \equiv -1 \pmod{p}$. См. задачу 9.14, адресованную именно этому случаю.

Пример 9.18

Рассмотрим простое число $p = 3581$, *которое сравнимо с* 5 *по модулю* 8. *Число* $c = 2177$ — *квадратичный вычет по модулю* p, *что можно проверить с помощью символа Лежандра, являющегося частным случаем символа Якоби.*

```
p = Prime[501]
Mod[p, 8] == 5
c = 2177;
JacobiSymbol[c, p]
```

```
3581
```

```
True
```

```
True
```

Решение сравнения $x^2 \equiv 2177 \pmod{p}$ *дается формулой* $m \equiv \pm 2177^{(p+3)/8} \equiv \pm 3100 \pmod{3581}$, *поскольку* $c^{(p-1)/4} \equiv 1 \pmod{p}$ *(в противном случае ответом должно быть* $\pm 2c(4c)^{(p-5)/8}$*).*

```
If[PowerMod[c, (p - 1) / 4, p] == 1,
        m = PowerMod[c, (p + 3) / 8, p],
        m = Mod[2 c * PowerMod[4 c, (p - 5) / 8, p], p]]
PowerMod[{m, -m}, 2, p]
```

```
3100
```

```
{2177, 2177}
```

Случай 3: $p \equiv 1 \pmod{8}$.

Быстрого алгоритма для решения требуемого сравнения не существует. Мы следуем [Rabi79]. В разд. A.4 вводятся множество QR квадратичных вычетов по модулю p и множество NQR квадратичных невычетов по модулю p.

Пусть r и s обозначают два решения $\pm m$ сравнения $x^2 \equiv c \pmod{p}$. Тогда $r+u$ и $s+u$ — два решения сравнения $(x-u)^2 - c \equiv 0 \pmod{p}$. Другими словами,

$$(x-u)^2 - c = (x-(r+u))(x-(s+u)) \tag{9.32}$$

над конечным полем $\mathbb{Z}_p$ $(= \mathrm{GF}(p))$. Так как $r \not\equiv s \pmod{p}$, это влечет, что элемент $(r+u)/(s+u)$ никогда не принимает значения 1. Поскольку отображение $u \mapsto (r+u)/(s+u)$ взаимно однозначно для $u \in \mathbb{Z}_p$, $u \neq -s$, мы заключаем, что

$$\{(r+u)/(s+u) | u \in \mathbb{Z}_p \setminus \{-s\}\} = \mathbb{Z}_p \setminus \{1\}. \tag{9.33}$$

Читатель может пожелать проверить это с помощью функций *Table*, *Mod*, *PowerMod* и *Union* из пакета "Mathematica".

```
p = 19; s = 9;
r = p - s;
s1 = Table[Mod[(r + u) * PowerMod[(s + u), -1, p], p],
        {u, 0, r - 1}]
s2 = Table[Mod[(r + u) * PowerMod[(s + u), -1, p], p],
        {u, r + 1, p - 1}]
s = Union[s1, s2]
```

```
{18, 3, 8, 9, 4, 16, 15, 7, 10, 0}
```

```
{2, 11, 14, 6, 5, 17, 12, 13}
```

```
{0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18}
```

Из равенства (9.33) и теоремы А.20 следует, что для половины допустимых значений u элемент $(r+u)/(s+u)$ будет лежать в QR$\cup\{0\}$, а для другой половины он будет лежать в NQR. В первом случае либо $u=-r$, либо (по теореме А.21) как $r+u$, так и $s+u$ будут элементами QR или оба будут лежать в NQR. Во втором случае ровно одно из чисел будет лежать в QR, тогда как другое — в NQR.

Свойство квадратичных вычетов по модулю простого числа, которое потребуется нам позже, дается формулой (А.16):

$$x^{(p-1)/2}-1=\prod_{u\in QR}(x-u).$$

Пример 9.19

Рассмотрим в качестве примера квадратичные вычеты по модулю 11. *Введем новую функцию:*

```
ListQuadRes[p_] :=
    Select[Range[p], JacobiSymbol[#1, p] == 1 &]
```

```
p = 11;
ListQuadRes[p]
```

```
{1, 3, 4, 5, 9}
```

Таким образом, числа $1,3,4,5,9$ *суть QR по модулю* 11. *Применим теперь функцию* PolynomialMod *из пакета "Mathematica":*

```
PolynomialMod[(x - 1) * (x - 3) * (x - 4) * (x - 5) *
(x - 9), 11]
```

```
10 + x^5
```

Это действительно равно x^5-1 *по модулю* 11.

Из проведенного выше обсуждения, в частности, из равенств (9.33) и (А.16) следует, что для случайно выбранного $u\in\mathbb{Z}_p\setminus\{-s\}$

$$\text{НОД}((x-u)^2-c,\,x(x^{(p-1)/2}-1)) \bmod p \tag{9.34}$$

будет равен

$$\begin{array}{rlll}
x-u-r, & \text{если} & u+r\in QR\cup\{0\} & \text{и} \quad u+s\in NQR,\\
x-u-s, & \text{если} & u+r\in NQR & \text{и} \quad u+s\in QR\cup\{0\},\\
1, & \text{если} & u+r\in NQR & \text{и} \quad u+s\in NQR\\
(x-u)^2-c, & \text{если} & u+r\in QR\cup\{0\} & \text{и} \quad u+s\in QR\cup\{0\}.
\end{array}$$

Приведенные выше доводы влекут за собой, что одна из первых двух возможностей будет встречаться с вероятностью $\frac{(p-1)/2}{p-1}=1/2$. Поэтому с вероятностью $1/2$ мы имеем нетривиальный делитель многочлена $(x-u)^2-c$. Так как u известно, можно также найти значение r или s.

Заметим, что в крайне неправдоподобном оставшемся случае, а именно, $u = -s$, выражение $(x-u)^2 - c$ сводится к $x^2 + 2s \cdot x$. Тогда НОД в (9.34) будет содержать делитель x, а другой делитель даст решение s.

Позже мы дадим пример на рассмотренный метод.

Ожидаемое число тех u, которые должны испытываться в приведенном алгоритме до нахождения решения сравнения $x^2 \equiv c \pmod{p}$, обратно к 1/2, т.е. равно 2. Обсуждение других методов вычисления квадратных корней по модулю простого числа заинтересованный читатель может найти в [Pera86][12].

□ Четыре решения

Финальный шаг в алгоритме дешифрования — это, конечно, применение китайской теоремы об остатках для комбинирования каждого из двух решений сравнения $x^2 \equiv c \pmod{p}$ с каждым из двух решений сравнения $x^2 \equiv c \pmod{q}$.

Пример 9.16 (часть 3)

Мы придерживаемся параметров примера 9.16. *Таким образом,* $p = 9733$, $q = 10177$, $n = p \times q = 99052741$, *а решения систем*

$$\begin{array}{rcll} a & \equiv & 1 \pmod{9733} & \\ b & \equiv & 0 \pmod{9733} & \end{array} \quad u \quad \begin{array}{rcll} a & \equiv & 0 \pmod{10177}, & \\ b & \equiv & 1 \pmod{10177}. & \end{array}$$

суть $a = 45287650$ *и* $b = 53765092$.

```
p = 9733; q = 10177; n = p * q;
a = 45287650; b = 53765092;
Mod[p, 8]
Mod[q, 8]
```

```
5
```

```
1
```

Пусть $c = 9513124$ — *шифртекст. Так как* $p \equiv 5 \pmod{8}$ *и* $q \equiv 1 \pmod{8}$, *мы следуем случаю 2 нахождения квадратных корней из* c *по модулю* p *и случаю 3 нахождения квадратных корней по модулю* q.

$\sqrt{9513124}$ по модулю p, случай 2

Вычисляем $c^{(p-1)/4} \bmod p$ *с помощью функции* PowerMod *из пакета "Mathematica":*

```
c = 9513124;
u = PowerMod[c, (p - 1) / 4, p]
```

```
1
```

[12] На русском языке в *[НодК99] можно найти изложение алгоритма Цассенхауза–Кантора и алгоритма Шэнкса.— *Прим. ред.*

и получаем 1. *Поэтому квадратный корень из* c *по модулю* p *равен* $\pm c^{(p+3)/8}$:

```
f = PowerMod[c, (p + 3) / 8, p]
```

|| 868

$\sqrt{9513124}$ по модулю q, случай 3

Мы хотим найти корни двучлена $x^2 - 9513124$ *по модулю* q. *Возьмем случайное* u *из* $\mathbb{Z}_q$ *и вычислим* НОД$((x-u)^2 - 9413124, x(x^{(q-1)/2}-1))$ *в надежде найти линейный делитель. При этом используется функция PolynomialGCD из пакета "Mathematica":*

```
u = 11; x = .;
PolynomialGCD[(x - u)^2 - c, x * (x^(q-1)/2 - 1, Modulus -> q]
```

|| 2492 + 10155 x + x^2

Еще одна попытка:

```
u = 111; x = .;
PolynomialGCD[(x - u)^2 - c, x*(x^(q-1)/2-1,Modulus -> q]
```

|| 1438 + x

Отсюда следует, что один из квадратных корней дается формулой $x - 111 - g \equiv x + 1438 \pmod{q}$, *где*

```
g = Mod[-111 - 1438, q]
```

|| 8628

По китайской теореме об остатках (теорема A.19) получаем четыре квадратных корня из сравнения $x^2 \equiv 9513124 \pmod{99052741}$:

```
Mod[a * f + b * g, n]
Mod[a * f - b * g, n]
Mod[-a * f + b * g, n]
Mod[-a * f - b * g, n]
```

|| 6969696

|| 63567091

|| 35485650

|| 92083045

9.5.3 Как различать решения

Пусть f — одно из двух решений сравнения $x^2 \equiv c \pmod{p_U}$, а g — одно из двух решений сравнения $x^2 \equiv c \pmod{q_U}$. Далее, пусть a и b — решения системы линейных сравнений (9.30) и (9.31). Тогда по китайской теореме об остатках (теорема А.19) четыре решения сравнения (9.29) даются формулой

$$\pm f \cdot a \pm g \cdot b \pmod{n_U}.$$

Отправителю и получателю следует различать эти четыре решения, чтобы они могли согласовать одно из них. В некоторых случаях это сделать легко. В самом деле, если оба числа p_U и q_U сравнимы с 3 по модулю 4, то по следствию А.24 число -1 является квадратичным невычетом (NQR) по обоим модулям p_U и q_U. Следовательно, в точности одно из чисел f или $-f$ является QR, и то же самое верно для g и $-g$. Заменяя f на $-f$ и/или g на $-g$, если нужно, мы без ограничения общности получаем, что

$+f \cdot a + g \cdot b$	есть QR $\bmod p_U$,	$+f \cdot a + g \cdot b$	есть QR $\bmod q_U$,
$+f \cdot a - g \cdot b$	есть QR $\bmod p_U$,	$+f \cdot a - g \cdot b$	есть QR $\bmod q_U$,
$-f \cdot a + g \cdot b$	есть QR $\bmod p_U$,	$-f \cdot a + g \cdot b$	есть QR $\bmod q_U$,
$-f \cdot a - g \cdot b$	есть QR $\bmod p_U$,	$-f \cdot a - g \cdot b$	есть QR $\bmod q_U$.

В силу определения А.11 и второго утверждения теоремы А.26 о символах Якоби мы имеем $((f \cdot a + g \cdot b)/n_U) = ((-f \cdot a - g \cdot b)/n_U) = 1$, тогда как $((f \cdot a - g \cdot b)/n_U) = ((-f \cdot a + g \cdot b)/n_U) = -1$. Из двух решений с символом Якоби $+1$ одно будет лежать между 1 и $(n_U - 1)/2$, другое — между $(n_U + 1)/2$ и $n_U - 1$ (или же оба равны 0). Заключаем, что существует единственное решение m, такое, что $0 \leq m \leq (n_U - 1)/2$ и $(m/n_U) = 1$. Таким образом, отправитель и получатель могут согласованно использовать сообщения только этой формы.

Пример 9.20 (часть 1)

Пусть $n_B = 77$, а $c = 53$ — полученное сообщение. Применяя процесс дешифрования, описанный в предыдущем подразделе, получаем $f = 2$, $g = 8$, $a = 22$ и $b = 56$.

С помощью функций Mod и JacobiSymbol из пакета "Mathematica" получаем следующие четыре возможных сообщения и соответствующие им значения символов Якоби.

```
nB = 77; f = 2; g = 8;
a = 22; b = 56;
m1 = Mod[a * f + b * g, nB];
m2 = Mod[a * f - b * g, nB];
m3 = Mod[-a * f + b * g, nB];
m4 = Mod[-a * f - b * g, nB];
```

```
Print[m1, " ", JacobiSymbol[m1, nB]]
Print[m2, " ", JacobiSymbol[m2, nB]]
Print[m3, " ", JacobiSymbol[m3, nB]]
Print[m4, " ", JacobiSymbol[m4, nB]]
```

```
|| 30  -1
|| 58  1
|| 19  1
|| 47  -1
```

Мы видим, что $m = 19$ *– единственное решение, такое, что* $(m/77) = 1$ *и* $0 \leq m \leq 38$, *поэтому* $m = 19$ *и было отправленным сообщением.*

Если p_U (или q_U) сравнимо с 1 по модулю 4, все еще можно согласованно использовать только такие сообщения m, что $0 \leq m \leq$ $\leq (n_U - 1)/2$. Для получения равенства $(m/n_U) = 1$ отправитель и получатель могут ограничиться более короткими сообщениями, скажем, короче 20 цифр, дополняя их другими цифрами так, чтобы результирующее сообщение имело символ Якоби, равный 1 по модулю n_U.

9.5.4 Эквивалентность взлома схемы Рабина и факторизации числа n

Теперь мы покажем, что взлом рабинского варианта схемы RSA эквивалентен факторизации n_U. Конечно, когда факторизация n_U известна криптоаналитику, система Рабина фактически взломана, так как криптоаналитик может использовать те же самые методы дешифрования, что и получатель (см. подразд. 9.5.2).

Теорема 9.18
Пусть $n = p \times q$, где p и q — простые числа. Пусть A обозначает алгоритм, который для любого c, являющегося квадратом целого числа, находит решение сравнения $x^2 \equiv c \pmod{n}$ с помощью $F(n)$ операций. Тогда существует вероятностный алгоритм, факторизующий n с ожидаемым числом операций $2(F(n) + 2\log_2 n)$.

Доказательство. Выберем случайное $m, 0 < m < n$, вычислим $c \equiv$ $\equiv m^2 \pmod{n}$ и решим сравнение $x^2 \equiv m \pmod{n}$ с помощью алгоритма A за $F(n)$ шагов. Пусть k — решение, найденное A. Каждая из следующих четырех возможностей имеет вероятность 1/4:

$$\begin{array}{llcll} 1) & k \equiv +m & (\mathrm{mod}\ p) & \text{и} & k \equiv +m \pmod{q}, \\ 2) & k \equiv +m & (\mathrm{mod}\ p) & \text{и} & k \equiv -m \pmod{q}, \\ 3) & k \equiv -m & (\mathrm{mod}\ p) & \text{и} & k \equiv +m \pmod{q}, \\ 4) & k \equiv -m & (\mathrm{mod}\ p) & \text{и} & k \equiv -m \pmod{q}. \end{array}$$

В самом деле, существуют четыре сообщения, которые отображаются в c, и все четыре одинаково правдоподобны.

В случае 2) НОД$(k-m,n)=p$, а в случае 3) НОД$(k-m,n)=q$. Таким образом, вычисление НОД$(k-m,n)$ дает разложение n с веротностью $1/2$. Это вычисление по теореме A.9 включает менее $2\log_2 n$ операций. Поэтому каждый выбор m требует самое большее $F(n)+2\log_2 n$ операций. Так как вероятность успеха равна $1/2$, в среднем нужны две попытки.

■

Пример 9.20 (часть 2)

Предположим, что $n=77$ и что выбранное значение m равно 30. Тогда $c=30^2\equiv 53 \pmod{77}$. Допустим теперь, что алгоритм A находит решение $k=19$ сравнения $x^2\equiv 53 \pmod{77}$ (см. эти параметры в примере 9.20 (часть 1)).

Тогда один из делителей числа n определяется из НОД$(k-m,n)$. *Это также произойдет, если A найдет решение $k=58$, но не с числами 30 или 47. Все вычисления легко проверяются посредством функции GCD из пакета "Mathematica".*

```
n = 77; m = 30;
GCD[19 - m, n]
GCD[58 - m, n]
GCD[30 - m, n]
GCD[47 - m, n]
```

```
11
7
77
1
```

9.6 Задачи

Задача 9.1. Рассмотрим систему RSA с $n=383\times 563$ (так что $n=215629$) и публичным ключом $e=49$. Значит, открытый текст m шифруется в $c=E(m)$, где

$$E(m)=m^{49} \bmod n.$$

Покажите, что любой шифртекст удовлетворяет сравнению $E^{10}(c)\equiv c \pmod{n}$. (Совет: используйте теорему Ферма и китайскую теорему об остатках.) Обозначение $E^{10}(c)$ стоит вместо $\overbrace{E(E(\ldots E}^{10}(c)\ldots))$. Дайте криптоаналитику легкий способ восстановить открытый текст m из шифртекста c.

Задача 9.2. Проверьте, что система шифрования (или схема подписи) RSA работает правильно, когда сообщение m имеет нетривиальный делитель,

общий с модулем $n = p \times q$, т.е. покажите, что

$$(m^e)^d \equiv m \pmod{n},$$

когда НОД$(m, n) = p$ или q (как всегда, e и d обозначают публичный и, соответственно, секретный показатели).

Задача 9.3. Рассмотрим криптосистему RSA с модулем $n = p \times q$ и публичным показателем e.

a) Докажите, что когда u делит $p - 1$, число решений уравнения $m^u \equiv 1 \pmod{p}$ есть в точности u (совет: используйте мультипликативную структуру GF(p), теорема B.20).

b) Покажите, что каждое решение уравнения $m^{e-1} \equiv 1 \pmod{p}$ является решением уравнения $m^{\text{НОД}(e-1,p-1)} \equiv 1 \pmod{p}$ и обратно (воспользуйтесь теоремой Ферма).

c) Докажите, что число решений уравнения $m^e \equiv m \pmod{p}$ дается выражением $1 + \text{НОД}(e-1, p-1)$.

d) Докажите, что число открытых текстов m, таких, что

$$m^e \equiv m \pmod{n}$$

(в этом случае шифрование не утаивает сообщение) равно

$$\{1 + \text{НОД}(e-1, p-1)\} \cdot \{1 + \text{НОД}(e-1, q-1)\}.$$

(Совет: используйте китайскую теорему об остатках.)

Задача 9.4. Продемонстрируйте принцип теста простоты Соловэя–Штрассена на числе $m = 33$. Число m в этой задаче взято небольшим, чтобы вычисления были просты. Не используйте числа, которые "случайно" имеют общий с m делитель.

Задача 9.5^M. Реализуйте в системе "Mathematica" алг. 9.14 и проверьте его для двух значений $m, 24 < m < 24^2$.

Задача 9.6^M. Дайте полное разложение числа $n = 110545695839248001$ с помощью ρ-алгоритма Полларда.

Задача 9.7^M. Завершите пример 9.7. (Совет: распространите поиск на интервал (-105, 105).)

Задача 9.8^M. Примените атаку Винера к $n = 6089471299$ и $e = 3097347557$.

Задача 9.9^M. Найдите сильного лгуна для составного числа $m = 85$.

Задача 9.10. Предположим, что Алиса посредством системы RSA посылает одно и то же сообщение субъектам B, C, D, E и F. Пусть публичные модули этих субъектов равны $n_B = 324059$, $n_C = 324371$, $n_D = 326959$, $n_E = 324851$ и $n_F = 324899$. Допустим, что все они имеют один и тот же публичный показатель $e = 5$.

Пусть перехваченные шифртексты равны $c_B = 68207$, $c_C = 96570$, $c_D = 251415$, $c_E = 273331$ и $c_F = 154351$. Определите сообщение Алисы (см. пример 9.8).

Задача 9.11. Предположим, что Алиса посредством системы RSA посылает Бобу секретные сообщения $m_1 = m$ и $m_2 = m^2 + 10m + 20$. Допустим, что модуль Боба $n_B = 483047$, а $e_B = 3$. Предположим, что Вы перехватили переданные шифртексты $c_1 = 346208$ и, соответственно, $c_2 = 230313$ и что Вам известно указанное выше соотношение между m_1 и m_2. Определите m_1 (см. пример 9.10).

Задача 9.12. Рассмотрим вариант Рабина системы RSA. Значит, публично только число n. Предположим, что было послано сообщение $m, 1 < m < n$, имеющее нетривиальный общий с n делитель. Сколько возможных открытых текстов найдет получатель, закончив процесс дешифрования?

Задача 9.13. Вариант Рабина системы RSA используется в качестве криптосистемы с $n = 17419 \times 17431$. Продемонстрируйте алгоритм дешифрования этой системы для шифртекста $c = 234279292$. Какими будут решения, если следовать методу подразд. 9.5.3? Почему этот метод может быть применен?

Задача 9.14. Пусть $p \equiv 5 \pmod 8$ и c — квадратичный вычет по модулю p.

a) Покажите, что $c^{(p-1)/4} \equiv \pm 1 \pmod p$.

b) Покажите, что в случае $c^{(p-1)/4} \equiv 1 \pmod p$ решением сравнения $x^2 \equiv c \pmod p$ служит $\pm c^{(p+3)/8}$.

c) Покажите, что в случае $c^{(p-1)/4} \equiv -1 \pmod p$ решением сравнения $x^2 \equiv c \pmod p$ служит $\pm 2c(4c)^{(p-5)/8}$. (Совет: используйте теорему А.25, из которой слелует, что число 2 не является квадратичным вычетом по модулю p.)

Глава 10

Системы, основанные на эллиптических кривых

В этой главе объясняется, что криптосистемы, основанные на проблеме дискретных логарифмов, могут также определяться над эллиптическими кривыми (EC). То же самое может быть сделано для систем, основанных на методе RSA, но резон делать это представлется незначительным. Для систем над эллиптическими кривыми, подобных системам дискретных логарифмов, допустимы очень малые значения параметров, обеспечивающие тот же уровень безопасности, какой дают обычные системы над конечными полями.

Однако многие вопросы, касающиеся EC-систем, на данный момент остаются еще открытыми, что делает неясным будущее этих систем.

10.1 Некоторые основные факты об эллиптических кривых

Пусть $\mathrm{GF}(q)$ — конечное поле из q элементов, где $q = p^m$. Число p — простое и называется характеристикой поля $\mathrm{GF}(q)$. Если $m = 1$, то $\mathrm{GF}(q) = \mathbb{Z}_p$, это множество целых чисел по модулю p.

Так называемое (аффинное) *уравнение Вейерштрасса* имеет вид

$$y^2 + u \cdot x \cdot y + v \cdot y = x^3 + a \cdot x^2 + b \cdot x + c. \tag{10.1}$$

Оно может определяться над любым полем (наподобие $\mathbb{R}$ или $\mathbb{C}$), но для криптографических целей мы всегда будем предполагать, что коэффициенты лежат в $\mathrm{GF}(q)$.

Если $p \neq 2$, то уравнение Вейерштрасса можно упростить посредством преобразования $y \mapsto y - (u \cdot x + v)/2$. Получится (с новыми значениями a, b и c) уравнение

$$y^2 = x^3 + a \cdot x^2 + b \cdot x + c. \tag{10.2}$$

Если, кроме того, $p \neq 3$, то можно применить также преобразование $x \mapsto x - a/3$, чтобы привести уравнение к виду

$$y^2 = x^3 + b \cdot x + c. \tag{10.3}$$

При $p = 2$ для соотношения (10.1) возможны два стандартных упрощения:

$$y^2 + x \cdot y = x^3 + a \cdot x^2 + c, \tag{10.4}$$

$$y^2 + v \cdot y = x^3 + b \cdot x + c. \tag{10.5}$$

Определение 10.1
Эллиптическая кривая $\mathcal{E}$ над полем GF(q) — это множество точек (x, y), удовлетворяющих уравнению (10.1), вместе с единственным элементом O, называемым *точкой в бесконечности*.

Проверить, лежит ли точка (u, v) на данной эллиптической кривой, скажем, $y^2 = x^3 + 2x + 3$ над $\mathbb{Z}_5$, довольно легко.

```
p = 5;
a = 0; b = 2; c = 3;
EC[x_, y_] = y^2 - x^3 - a * x^2 - b * x - c;
{u, v} = {1, 4};
Mod[EC[u, v], p] == 0
```

```
True
```

Чтобы увидеть, содержит ли кривая $\mathcal{E}$ точку с заданной x-координатой, можно использовать функцию Solve из пакета "Mathematica". Поскольку уравнение Вейерштрасса квадратично по y, получатся самое большее два значения y (см. теорему B.14).

```
p = 11;
Solve[{y^2 == x^3 - 5*x + 3, x == 3, Modulus == p}, {y}]
```

```
{{Modulus→11,x→3,y→2},{Modulus→11,x→3,y→9}}
```

Таким образом, $x = 3$ приводит к значениям $y = \pm 2$, т.е. к точкам $(3, 2)$ и $(3, 9)$. Читателю стоило бы испытать некоторые другие значения переменной x.

При этом читатель может обратиться к подразд. 9.5.2, где он найдет обсуждение вопроса о том, как математическими средствами найти квадратные корни из квадратичного вычета по модулю простого числа.

Из вышесказанного следует, что точка $P = (x, y)$ на эллиптической кривой полностью характеризуется координатой x и "знаком" при y. Если $q = p, p > 2$, то "знак" y можно определить как "плюс" при условии $0 \le y \le (p-1)/2$ и как "минус" в противном случае.

Если $q = p^m, p > 2$, то в качестве "знака" y можно использовать "знак" самой левой ненулевой координаты в p-ичном представлении y.

Для малых значений p можно определить все точки на $\mathcal{E}$, испытывая все значения x и проверяя в каждом случае, будет ли уравнение (10.1)

иметь решение. Ниже мы используем функции *Flatten*, *Table* и *Solve* из пакета "Mathematica".

```
Clear[x, y];
p = 11;
Flatten[Table[Solve[
    {y^2 == x^3 - 5*x + 3, x == u, Modulus == p}],
    {u, 0, p - 1}], 1]
```

```
{{Modulus→11,y→5,x→0},  {Modulus→11,y→6,x→0},
 {Modulus→11,y→1,x→2},  {Modulus→11,y→10,x→2},
 {Modulus→11,y→2,x→3},  {Modulus→11,y→9,x→3},
 {Modulus→11,y→5,x→4},  {Modulus→11,y→6,x→4},
 {Modulus→11,y→2,x→5},  {Modulus→11,y→9,x→5},
 {Modulus→11,y→5,x→7},  {Modulus→11,y→6,x→7},
 {Modulus→11,y→4,x→9},  {Modulus→11,y→7,x→9}}
```

Мы видим, что при $p = 11$ существуют 14 решений. Имеется (неточное) вероятностное рассуждение, предсказывающее число точек на $\mathcal{E}$: для каждого значения x уравнение (10.1) имеет два решения с вероятностью $1/2$ и вовсе не имеет решений с такой же вероятностью $1/2$, что приводит приблизительно к q решениям.

Чтобы подтвердить правдоподобность этого утверждения, рассмотрим правую часть уравнения (10.2), предполагая, что $p > 2$. Если для данного значения x правая часть является ненулевым квадратом в $\mathrm{GF}(p)$ (а имеется $(p-1)/2$ таких квадратов, именно все четные степени примитивного элемента из $\mathrm{GF}(p)$; или же см. теорему A.20), то y будет иметь два решения. Если правая часть равна 0, то имеется единственное решение $y = 0$. Никаких других решений нет.

Известная теорема Хассе [Silv86] гласит:

Теорема 10.1 (*Хассе*)
Пусть N — число точек на эллиптической кривой над $\mathrm{GF}(q)$. Тогда

$$|q + 1 - N| \leq 2\sqrt{q}.$$

Отметим, что в примере выше мы действительно имеем $|12 - 14| \leq 2\sqrt{11}$.

Вообще говоря, очень трудно найти точное число точек на эллиптической кривой. Однако имеется алгоритм Шуфа [Scho95], вычисляющий это число (для дальнейшего обсуждения см. также [Mene93]).

Мы хотим напомнить читателю возможности пакета "Mathematica" в области вычислений над полем $\mathrm{GF}(p^m)$, где $m > 1$, хотя для понимания оставшейся части главы это не является необходимым.

Пример 10.1

В качестве примера кривой над $\mathrm{GF}(2^4) = \mathrm{GF}(2)[\alpha]/(1 + \alpha^3 + \alpha^4)$

(см. табл. В.2) рассмотрим уравнение $y^2 = x^3 + \alpha x + 1$. *Чтобы проверить, лежит ли на этой кривой точка* (α^2, α^{14}), *загрузим сначала пакет Algebra'FiniteFields' из "Mathematica".*

```
<< Algebra'FiniteFields'
```

```
f16 = GF[2, {1, 0, 0, 1, 1}];
al = f16[{0, 1, 0, 0}];
EC[x_, y_] = y^2 - x^3 - al * x - 1;
{u, v} = {al^2, al^14};
EC[u, v]
```

```
|| 0
```

В самом деле, $(\alpha^{14})^2 = (\alpha^2)^3 + \alpha(\alpha^2) + 1$, *что можно легко проверить:*

```
al^6 + al^3 + 1
(al^14)^2
```

```
|| {0, 1, 1, 0}_2
|| {0, 1, 1, 0}_2
```

10.2 Геометрия эллиптических кривых

Причиной нашего интереса к эллиптическим кривым служит операция сложения, которая может быть на них определена. Относительно этой операции точка в бесконечности $O \in \mathcal{E}$ служит нулем, а множество $\mathcal{E}$ обладает структурой аддитивной группы.

Чтобы определить сложение на $\mathcal{E}$, воспользуемся тем свойством, что любая прямая, пересекающая $\mathcal{E}$ по меньшей мере в двух точках, пересекает ее трижды. Точка касания считается здесь за две точки. Точка O в бесконечности — это точка пересечения всех "вертикальных" прямых.

Приведем сначала рисунок эллиптической кривой над полем действительных чисел. Мы используем функцию *ImplicitPlot* из "Mathematica" для чего сначала нужно загрузить пакет *Graphics'ImplicitPlot'*.

```
<< Graphics'ImplicitPlot'
```

```
elliptic = ImplicitPlot[y^2 == x^3 - 5 * x + 3, {x, -3, 3}]
```

Полученный результат приведен на рис. 10.1.

Мы призываем читателя заменить коэффициент -5 при x у функции, изображенной выше, на -4, а затем на -3 и посмотреть, как изменится график.

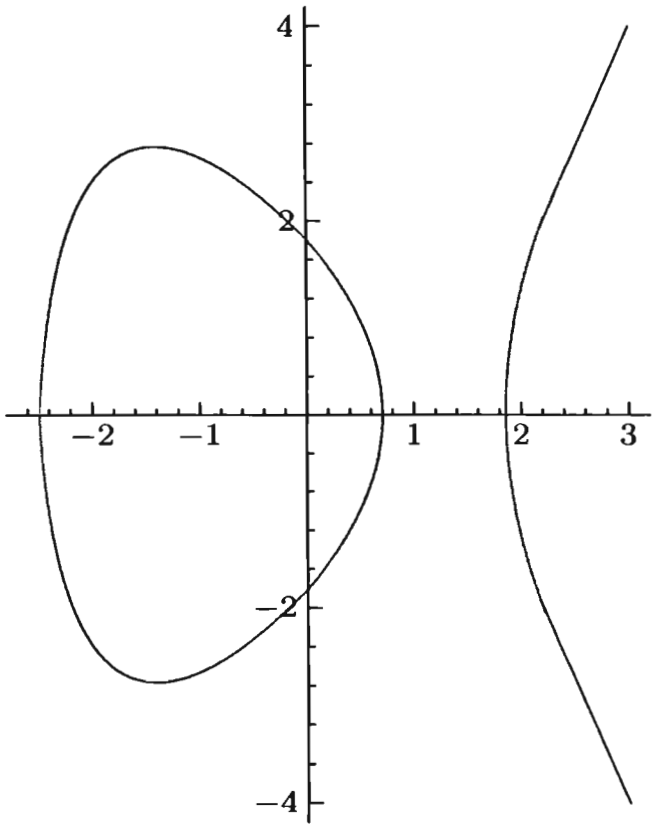


Рис. 10.1. Эллиптическая кривая над полем действительных чисел.

Чтобы увидеть, где прямая $y = -x + 1$ пересекает кривую $y^2 = x^3 - 5x + 3$, используем дополнительные функции *Epilog* и *Line*.

```
ImplicitPlot[y^2 == x^3 - 5 * x + 3, {x, -3, 3},
    PlotRange -> {-4, 4},
    Epilog -> Line[{{-3, 4}, {4, -3}}]]
```

Полученный результат приведен на рис. 10.2.

Чтобы численно найти точки пересечения, можно воспользоваться функцией *NSolve*.

```
NSolve[{y^2 == x^3 - 5 * x + 3, y == -x + 1}, {x, y}]
```

```
{{y→-1., x→2.}, {y→0.381966, x→0.618034},
{y→2.61803, x→-1.61803}}
```

Когда кривая определена над $\mathbb{Z}_p$, точки ее пересечения с прямой линией можно найти посредством функции *Solve* следующим образом.

```
p = 11;
Solve[{y^2 == x^3 - 5 * x + 3, y == x - 1, Modulus == p},
        {x, y}]
```

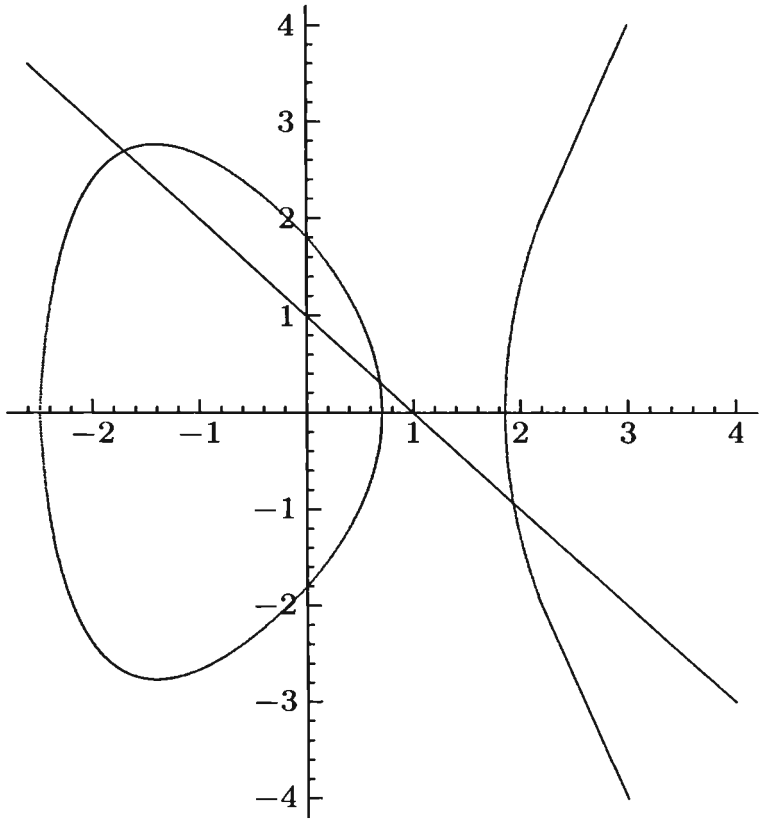


Рис. 10.2. Пересечение $y = -x + 1$ и $y^2 = x^3 - 5x + 3$.

```
{{Modulus→11, y→1, x→2}, {Modulus→11, y→2, x→3},
{Modulus→11, y→6, x→7}}
```

Иной способ нахождения точек пересечения прямой $y = u \cdot x + v$ с эллиптической кривой — подставить $y = u \cdot x + v$ в уравнение (10.1), получая уравнение третьей степени относительно x и находя затем разложение соответствующего многочлена.

Пример 10.2

Допустим, что мы работаем над $\mathbb{Z}_{11}$. *Чтобы найти точки пересечения прямой* $y = 4x + 1$ *с кривой* $y^2 = x^3 - 5x + 3$, *разложим многочлен* $(4x + 1)^2 - (x^3 - 5x + 2)$ *с помощью функции* Factor *из пакета "Mathematica":*

```
p = 11;
Clear[x];
ec = x^3 - 5 * x + 3;
il = 4 * x + 1;
Factor[il^2 - ec, Modulus -> p]
```

```
10 (2 + x) (7 + x) (8 + x)
```

Получаем x*-значения точек пересечения:* -2, -7 *и* -8. *Исходя из равенства* $y = 4x + 1$, *находим решения* $(9, 4)$, $(4, 6)$ *и* $(3, 2)$:

```
x = Mod[{-2, -7, -8}, p]
y = Mod[4 * x + 1, p]
```

```
{9, 4, 3}
{4, 6, 2}
```

□ Прямая, проходящая через две различные точки

Пусть $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ — две различные точки на эллиптической кривой $\mathcal{E}$ (обе не в бесконечности). Пусть $\mathcal{L}$ — прямая, проходящая через P_1 и P_2. Как найти третью точку пересечения $\mathcal{L}$ с $\mathcal{E}$? Если $x_1 = x_2$ и $y_1 = -y_2$, то[1] по определению третьей точкой пересечения будет O.

Поэтому рассмотрим случай $x_1 \neq x_2$. Прямая $\mathcal{L}$, проходящая через P_1 и P_2, задается уравнением

$$y - y_1 = \lambda(x - x_1), \quad \text{где} \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1}. \tag{10.6}$$

Обсудим два случая.

Случай $p \neq 2$

Допустим, что эллиптическая кривая уже записана в приведенной форме (10.2). Подстановка (10.6) в это равенство дает $(\lambda(x - x_1) + y_1)^2 = x^3 + a \cdot x^2 + b \cdot x + c$. Мы знаем два корня уравнения третьей степени и должны найти третий корень (назовем его x_3). Таким образом, данное уравнение можно записать в виде $(x - x_1)(x - x_2)(x - x_3) = 0$. Сравнивая коэффициенты при x^2 в обоих выражениях, получаем

$$x_3 = \lambda^2 - a - x_1 - x_2 \tag{10.7}$$

и в силу (10.6)

$$y_3 = \lambda(x_3 - x_1) + y_1. \tag{10.8}$$

Пример 10.3

Рассмотрим эллиптическую кривую $y^2 = x^3 + 11x^2 + 17x + 25$ *над* $\mathbb{Z}_{31}$. *Точки* $P_1 = (x_1, y_1) = (2, 7)$ *и* $P_2 = (x_2, y_2) = (23, 9)$ *лежат на* $\mathcal{E}$, *что можно проверить с помощью функции* Mod:

```
p = 31;
a = 11; b = 17; c = 25;
x1 = 2; y1 = 7; x2 = 23; y2 = 9;
F[x_, y_] := y^2 - (x^3 + a * x^2 + b * x + c);
Mod[F[x1, y1], p] == 0
Mod[F[x2, y2], p] == 0
```

[1] Здесь, как и ниже, предполагается, что кривая задана уравнением (10.2).— *Прим. ред.*

```
|| True

|| True
```

Наклон λ *прямой* $\mathcal{L}$, *проходящей через* P_1 *и* P_2, *дается в* (10.6): $\lambda = \frac{9-7}{23-2} = 2 \times 3 = 6$. *Мы используем здесь функцию* PowerMod, *чтобы получить мультипликативное обратное к* 21 *по модулю* 31.

```
PowerMod[21, -1, p]
```

```
|| 3
```

Координаты (x_3, y_3) *третьей точки пересечения* $\mathcal{L}$ *с* $\mathcal{E}$ *даются формулами* (10.7) *и* (10.8):

```
lam = 6;
x3 = Mod[lam^2 - a - x1 - x2, p]
y3 = Mod[lam * (x3 - x1) + y1, p]
```

```
|| 0

|| 26
```

Точка $P_3 = (0, 26)$ *действительно лежит на* $\mathcal{E}$, *что можно проверить вычислением:*

```
Mod[F[x3, y3], p] == 0
```

```
|| True
```

Случай $p = 2$

Допустим теперь, что мы привели уравнение (10.1) к виду (10.4). Как и выше, подставляем соотношение (10.6) в уравнение (10.4) и сравниваем коэффициенты при x^2. Получаем

$$x_3 = \lambda^2 + \lambda - a - x_1 - x_2, \tag{10.9}$$

$$y_3 = \lambda(x_3 - x_1) + y_1. \tag{10.10}$$

Заметим, что все знаки "минус" можно заменить знаками "плюс", так как $p = 2$.

□ Касательная прямая

Имеется еще одна возможность, которую мы должны обсудить, а именно, когда $P_1 = (x_1, y_1) = P_2 = P$. Пусть $\mathcal{L}$ — *касательная прямая* к $\mathcal{E}$, проходящая через P. Это означает, что $\mathcal{L}$ пересекает $\mathcal{E}$ в точке $P = (x_1, y_1)$ и наклон у $\mathcal{L}$ совпадает с производной [y по x относительно $\mathcal{E}$ — *Ред.*] в точке P. Обычно P рассматривается как точка пересечения с кратностью два.

Над $\mathbb{R}$ ситуация выглядит так:

```
ImplicitPlot[y^2 == x^3 - 5x - 3, {x, -2, 4},
      PlotRange -> {-4, 4},
      Epilog -> Line[{{-2, 2}, {4, -4}}]]
```

Полученный результат приведен на рис. 10.3.

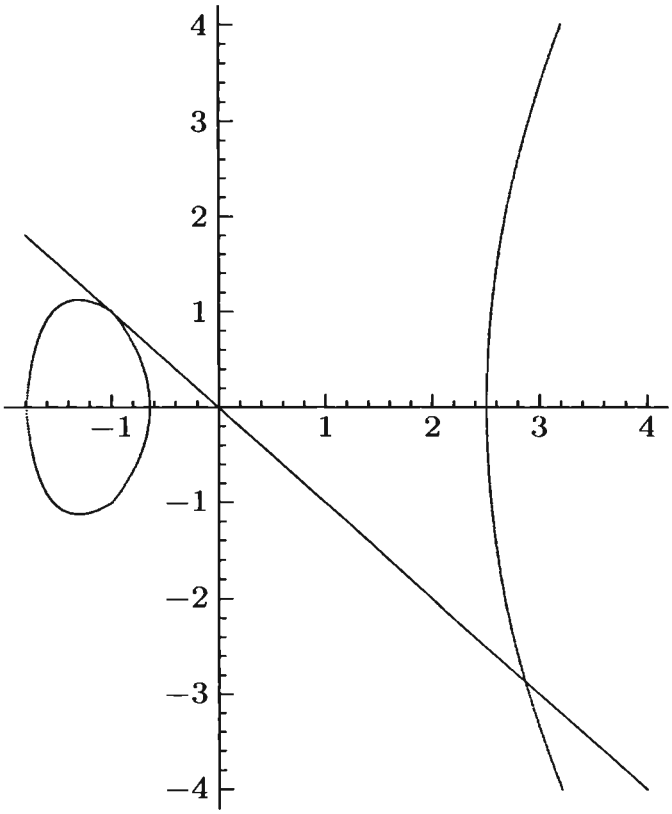


Рис. 10.3. Касательная к эллиптической кривой.

Мы здесь пока исключили возможность того, что $\mathcal{L}$ — двойная касательная прямая к $\mathcal{E}$ (т.е., что кратность точки P равна 3). Если бы было именно так, то касательная прямая уже пересекала бы $\mathcal{E}$ в точке с кратностью 3. В дальнейшем, говоря о взятии *производной* многочлена над конечным полем, мы имем в виду формальное дифференцирование и последующее приведение коэффициентов по модулю характеристики поля. Например, в $\mathrm{GF}(3^m)$ производная от $x^4+2x^3+x^2+1$ равна $4x^3+6x^2+2x$, что приводится к x^3+2x.

Случай $p \neq 2$

Наклон касательной прямой, проходящей через точку $P = (x_1, y_1)$ на кривой $y^2 = x^3 + a \cdot x^2 + b \cdot x + c$ (см. (10.2)), задается значением y', определенным неявным дифференцированием, т.е. из равенства $2y_1 \cdot y' = 3x_1^2 + 2a \cdot x_1 + b$. Мы заключаем, что касательная прямая, проходящая

через P, дается уравнением

$$y - y_1 = \lambda(x - x_1), \quad \text{где} \quad \lambda = \frac{3x_1^2 + 2ax_1 + b}{2y_1}. \tag{10.11}$$

Для нахождения третьей точки пересечения прямой $\mathcal{L}$ и кривой $\mathcal{E}$ можно вновь воспользоваться равенствами (10.7) и (10.8).

Случай $p = 2$

Наклон касательной прямой, проходящей через точку P на кривой $y^2 + x \cdot y = x^3 + a \cdot x^2 + c$ (см. (10.4)), задается значением y', определяемым из равенства $2y_1 \cdot y' + y_1 + x_1 \cdot y' = 3x_1^2 + 2a \cdot x_1$ или $y_1 + x_1 \cdot y' = x_1^2$. Следовательно, касательная прямая, проходящая через P, дается уравнением

$$y - y_1 = \lambda(x - x_1), \quad \text{где} \quad \lambda = \frac{x_1^2 + y_1}{x_1} = x_1 + \frac{y_1}{x_1}. \tag{10.12}$$

Чтобы найти найти третью точку пересечения $\mathcal{L}$ с $\mathcal{E}$, заметим, что равенство (10.9) сводится к

$$x_3 = a - \lambda^2 - \lambda = a + x_1^2 + \left(\frac{y_1}{x_1}\right)^2 + x_1 + \frac{y_1}{x_1} =$$

$$= a + x_1^2 + x_1 + \frac{y_1^2 + x_1 y_1}{x_1^2} \overset{(10.4)}{=} a + x_1^2 + x_1 + \frac{x_1^3 + ax_1^2 + c}{x_1^2},$$

т.е. к соотношению

$$x_3 = x_1^2 + \frac{c}{x_1^2}, \tag{10.13}$$

и что (10.10) приводит к равенству

$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3. \tag{10.14}$$

Пример 10.4

Рассмотрим эллиптическую кривую $y^2 + x \cdot y = x^3 + \alpha^9 x^2 + \alpha$ *над* GF(16), *где* $\alpha^4 = \alpha + 1$. *Точка* (α^2, α^{12}) *лежит на этой кривой, что легко проверить, загрузив сначала пакет Algebra'FiniteFields' из "Mathematica".*

```
<< Algebra'FiniteFields'
```

```
f16 = GF[2, {1, 1, 0, 0, 1}];
a = f16[{0, 1, 0, 0}];
EC[x_, y_] = y^2 + x * y - x^3 - a^9 * x^2 - a;
{x1, y1} = {a^2, a^12};
EC[x1, y1]
```

```
0
```

Касательная, проходящая через точку (α^2, α^{12}), *имеет наклон* λ, *определяемый равенством* (10.12). *Поэтому*

```
lam = x1 + y1 / x1
```

|| {1, 1, 0, 0}$_2$

что равно α^4. *Чтобы найти другую точку, где касательная пересекает* $\mathcal{E}$, *используем соотношения* (10.13) *и* (10.14):

```
x3 = x1^2 + a / x1^2
y3 = x1^2 + (x1 + y1 / x1) * x3
```

|| {0, 0, 1, 1}$_2$

|| {0, 0, 1, 0}$_2$

Таким образом, $(x_3, y_3) = (\alpha^6, \alpha^2)$. *Все это можно легко проверить:*

```
x3 = a^6
y3 = a^2
EC[x3, y3]
```

|| {0, 0, 1, 1}$_2$

|| {0, 0, 1, 0}$_2$

|| 0

10.3 Сложение точек на эллиптической кривой

В предыдущем разделе было показано, что прямая, проходящая через две точки на эллиптической кривой, пересекает эту кривую в третьей точке, а также показано, каким образом можно эффективно вычислить эту точку. То же самое выполняется для прямой, которая касается $\mathcal{E}$, с уточнением, что точка касания считается дважды.

Теперь мы готовы определить сложение на $\mathcal{E}$. Геометрическая идея, лежащая в основе приводимых ниже формул, состоит в следующем. Прежде всего, если $P = (x, y)$ — точка на эллиптической кривой, определенной уравнением (10.1), то полагаем

$$-P = (x, -y - u \cdot x - v).$$

Если $u = v = 0$, как в уравнении (10.2), то это сводится к

$$-P = (x, -y).$$

Геометрически это можно описать следующим образом: вычисляется прямая $\mathcal{L}$, проходящая через O и P; она пересекает $\mathcal{E}$ в третьей точке, а именно, в точке $-P$. Как отмечено раньше, точка O в бесконечности должна интерпретироваться как точка пересечения всех вертикальных прямых.

Чтобы сложить точки P_1 и P_2, когда обе они не лежат в бесконечности, нужно выполнить следующие два шага:

1) вычислить прямую, проходящую через P_1 и P_2 (или, в случае $P_1 = P_2$, касательную прямую, проходящую через P_1), и найти третью точку пересечения с $\mathcal{E}$; пусть это будет Q;

2) найти сумму $P_1 + P_2$, определяемую равенством $P_3 := -Q$.

Точка O служит нейтральным элементом этого сложения и противоположна самой себе.

Определение 10.2 (*сложение*)
Пусть P — точка на эллиптической кривой $\mathcal{E}$ (определяемой уравнением (10.1)), O — точка в бесконечности. Тогда положим

$$P + O = O + P = P.$$

Далее, пусть $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ — две точки на $\mathcal{E}$, обе отличные от O.
Тогда сумма $P_3 = P_1 + P_2$ определяется равенствами

1) $P_3 = -Q$, если $x_1 \neq x_2$;
здесь Q — третья точка пересечения с $\mathcal{E}$ прямой $\mathcal{L}$, проходящей через P_1 и P_2.

2) $P_3 = -Q$, если $P_1 = P_2$ и касательная, проходящая через P_1, единична; здесь Q — третья точка пересечения с $\mathcal{E}$ данной касательной.

3) $P_3 = -P_1$, если $P_1 = P_2$ и проходящая через P_1 касательная — двойная.

4) $P_3 = O$, если $P_1 = -P_2$.

Заметим, что случай 3) можно интерпретировать как специальный вариант случая 2).

Изобразим два наиболее типичных случая, а именно, 1) и 2), привлекая эллиптические кривые над полем действительных чисел. Нам снова потребуется пакет *Graphics`ImplicitPlot`*.

```
<< Graphics`ImplicitPlot`
```

```
ImplicitPlot[y^2 == x^3 - 5 * x + 3, {x, -3, 4},
  Epilog -> {line[{{-3, -2}, {4, 5}}],
     Line[{{3, -6}, {3, 6}}],
```

```
    Text[
"\!\(\* StyleBox[\(P\), FontColor->RGBColor[
0, 0, 1]]\)\!\(\* StyleBox[\(+\), FontColor->
RGBColor[0, 0, 1]]\)\!\(\* StyleBox[\Q\),
FontColor->RGBColor[0, 0, 1]]\)", {2.3, -4}],
    Text["\!\(\* StyleBox[\(P\), FontColor->
RGBColor[0, 0, 1]]\)", {-2.4, -2.1}],
    Text["\!\(\* StyleBox[\(Q\), FontColor->
RGBColor[0, 0, 1]]\)", {0.35, 1.9}],
    Text["•", {-2.31, -1.4}],
    Text["•", {0.28, 1.26}],
    Text["•", {3.01, -3.9}]}]
```

Полученный результат приведен на рис. 10.4.

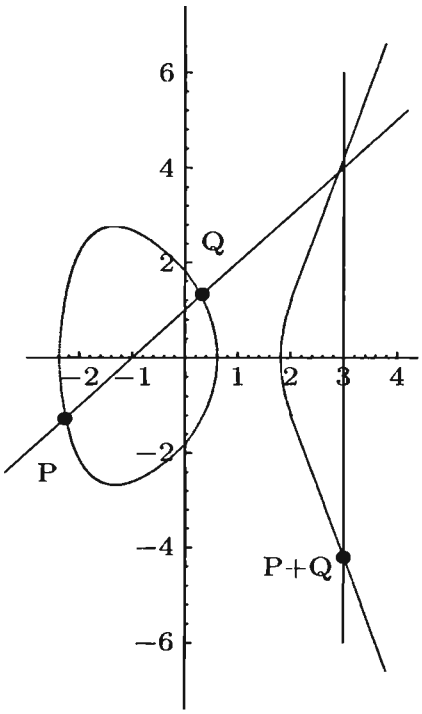


Рис. 10.4. Сложение точек на эллиптической кривой (первый случай).

```
ImplicitPlot[y^2 == x^3 - 5 * x - 3, {x, -2, 4},
  Epilog -> {line[{{3, -6}, {3, 6}}],
    Line[{{-2, 2}, {4, -4}}],
    Text["\!\(\* StyleBox[\(P\),
  FontColor->RGBColor[0, 0, 1]]\)", {-1.1, 1.65}],
    Text["\!\(\* StyleBox[\(2\), FontColor->
  RGBColor[0, 0, 1]]\)\!\(\* StyleBox[\(P\),
  FontColor->RGBColor[0, 0, 1]]\)", {2.8, 3.5}],
    Text["•", {-1.1, 1.1}],
    Text["•", {3.01, 3}]
```

Полученный результат приведен на рис. 10.5.

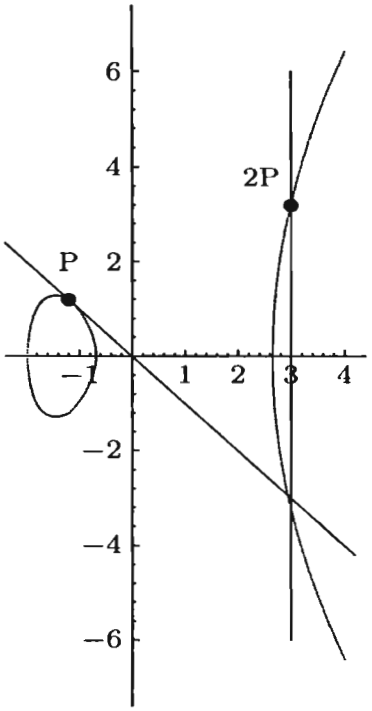


Рис. 10.5. Сложение точек на эллиптической кривой (второй случай).

Точки на эллиптической кривой вместе с определенным выше сложением образуют аддитивную группу. Мы не будем здесь доказывать это. Читатель отсылается к [Mene93] или [Silt92]. Отметим, что единственная нетривиальная часть доказательства — проверка ассоциативности сложения.

Теорема 10.2
Точки на эллиптической кривой $\mathcal{E}$ вместе со сложением, заданным в определении 10.2, образуют аддитивную группу. Нулевым элементом в ней служит точка O.

С помощью следующего модуля *Module* можно вычислять сумму двух точек на эллиптической кривой над полем GF(q) характеристики $p > 2$ (точка O в бесконечности будет обозначаться через $\{O\}$). Мы можем использовать формулы (10.6), (10.7), (10.8), (10.11) и применить функцию *Which* из пакета "Mathematica" с тем же порядком случаев, какой дан в определении 10.2.

```
EllipticAdd[p_, a_, b_, c_, P_List, Q_List] :=
Module[{lam, x3, y3, R},
  Which[P == {0}, R = Q,
    Q == {0}, R = P,
    P[[1]] != Q[[1]],
      lam = Mod[
    (Q[[2]] - P[[2]])*PowerMod[Q[[1]] - P[[1]], -1, p], p];
```

```
      x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
      y3 = Mod[-(lam * (x3 - P[[1]]) + P[[2]]), p];
      R = {x3, y3},
    (P == Q) && (P != {0}),
      lam = Mod[(3 * P[[1]]^2 + 2 * a * P[[1]] + b) *
    PowerMod[2 * P[[2]], -1, p], p];
      x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
      y3 = Mod[-(lam * (x3 - P[[1]]) + P[[2]]), p];
      R = {x3, y3},
    (P[[1]] == Q[[1]]) && (P[[2]] != Q[[2]]), R = {0}];
  R]
```

Покажем сложение точек в ряде случаев:

```
p = 11; a = 0; b = 6; c = 3;
EllipticAdd[p, a, b, c, {4, 6}, {9, 4}]
EllipticAdd[p, a, b, c, {9, 4}, {9, 4}]
EllipticAdd[p, a, b, c, {4, 6}, {4, 6}]
EllipticAdd[p, a, b, c, {4, 6}, {0}]
EllipticAdd[p, a, b, c, {4, 6}, {4, 5}]
EllipticAdd[p, a, b, c, {0}, {9, 4}]
```

```
|| {3, 9}
|| {7, 6}
|| {4, 5}
|| {4, 6}
|| {0}
|| {9, 4}
```

Заметим, что касательная, проходящая через точку (4,6) — двойная, поэтому определение 10.2,3) дает

$$(4,6) + (4,6) = -(4,6) = (4,5).$$

Как и в любой аддитивной группе, $2P$ обозначает $P + P$, $3P$ обозначает $P + P + P$ и т.д. Аналогично, $0P$ обозначает O, а $-n \cdot P$ обозначает $-(n \cdot P)$. Эти кратные точки P часто называют *скалярными кратными* P.

Порядок точки P — это наименьшее натуральное число n, для которого $n \cdot P = O$. Так как группа $\mathcal{E}$ конечна, это понятие вполне определено. Множество $\{O, P, 2P, \ldots, (n-1)P\}$ — циклическая подгруппа в $\mathcal{E}$. Отсюда следует, что n делит $|\mathcal{E}|$ (см. теорему B.5).

Теперь у нас есть модуль *EllipticAdd*, определенный выше, который позволяет легко вычислять $n \cdot P, n \geq 1$, с помощью рекурсии:

```
p = 11; a = 0; b = 6; c = 3; P = {9, 4};
f[1] = P;
f[n_] := f[n] = EllipticAdd[p, a, b, c, P, f[n - 1]];
Table[f[n], {n, 1, 5}] // ColumnForm
```

```
{9, 4}
{7, 6}
{7, 5}
{9, 7}
{0}
```

Таким образом, на кривой $y^2 = x^3 + 6x + 3$ над $\mathbb{Z}_{11}$ точка $P = (9, 4)$ имеет порядок 5.

В следующем разделе нам будет важно иметь точки на эллиптической кривой $\mathcal{E}$, порядок которых очень велик. Если мощность $\mathcal{E}$ известна и имеет специальный вид, например, $|\mathcal{E}|$ — малое кратное большого простого делителя, то довольно легко найти точки на $\mathcal{E}$ с известным большим порядком.

В качестве примера рассмотрим $|\mathcal{E}| = 3 \times 7919 = 23757$, и предположим, что $3P \neq O$. Тогда P имеет порядок либо 7919, либо 23757. Если $7919P = O$, то P имеет порядок 7919, в противном случае этот порядок будет иметь точка $3P$. Чтобы проверить эти утверждения, примените лемму B.4 и теорему B.5 (переписав мультипликативные обозначения на используемые здесь аддитивные).

10.4 Криптосистемы, определяемые над эллиптическими кривыми

Многие понятия в этом разделе можно рассматривать как прямой перевод понятий, введенных в гл. 8, но теперь в качестве главной операции вместо модулярного умножения используется сложение на эллиптической кривой. Модулярное возведение в степень переходит в скалярное умножение. По этой причине часто достаточно представить новые формулировки, не копируя всех доказательств.

В [Demy94] описана криптосистема, подобная RSA, но определенная над эллиптическими кривыми. Однако для взлома такой системы достаточно факторизовать ее модуль. Поскольку исходная система RSA имеет те же самые ограничения безопасности, но быстрее в вычислениях, резон использовать обобщение RSA на эллиптические кривые представляется слишком незначительным.

10.4.1 Проблема дискретных логарифмов над эллиптическими кривыми

В разд. 10.3 мы видели, как складываются точки на эллиптической кривой. Это операция с относительно низкой сложностью. Чтобы вычис-

лить скалярное кратное точки P скажем, $n \cdot P$ для некоторого целого n, можно использовать повторное сложение, но гораздо эффективнее скопировать идеи из подразд. 8.1.1.

Пример 10.5

Возьмем $n = 171$. Его двоичное представление — 10101011, *как показывает функция IntegerDigits из пакета "Mathematica":*

```
IntegerDigits[171, 2]
```

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

Поэтому для вычисления $171P$ достаточно вычислить

$$\begin{aligned} 2P &= P + P, \\ 4P &= 2P + 2P, \\ 8P &= 4P + 4P, \\ &\vdots \\ 128P &= 64P + 64P \end{aligned}$$

и сложить подходящие члены. Это можно сделать "на лету" следующим образом:

```
Clear[P];
2 ( 2 ( 2 ( 2( 2( 2( 2 P) + P)) + P)) + P) + P
```

```
171 P
```

Заметим, что мы только прибавляли частичные результаты к самим себе или к P. (Читатель может взглянуть на пример 8.3, где решается аналогичная задача модулярной арифметики.)

Конечно, аддитивные цепочки могут и дальше уменьшить сложность этих вычислений.

Проблема, обратная к проблеме вычисления скалярных кратных точки, состоит в следующем.

Определение 10.3
Пусть $\mathcal{E}$ — эллиптическая кривая, P — точка на $\mathcal{E}$, а Q — скалярное кратное P. *Проблема дискретных логарифмов* над эллиптической кривой — это проблема определения числа n из равенства

$$n \cdot P = Q. \tag{10.15}$$

Хотя известны методы решения уравнения (10.15), более эффективные, чем простое испытание $n = 1, 2, \ldots$, все эти методы имеют сложность порядка $n^{\alpha}, \alpha > 0$, и поэтому они экспоненциально медленнее, чем вычисление $n \cdot P$ из P, имеющее логарифмическую сложность.

10.4.2 Система дискретных логарифмов над эллиптическими кривыми

Теперь мы можем описать аналог протокола Диффи–Хеллмана обмена ключами (см. подразд. 8.1.2).

В качестве системных параметров нужны эллиптическая кривая $\mathcal{E}$ над конечным полем $\mathrm{GF}(q)$ и точка P на ней высокого порядка; например, порядок n точки P может иметь длину 150–180 цифр.

Каждый пользователь U системы выбирает секретный скаляр m_U, вычисляет точку $Q_U = m_U P$ и объявляет Q_U публично. Теперь Алиса и Боб могут согласовать общий ключ $K_{A,B} = m_A m_B P$. Алиса может найти этот общий ключ, вычисляя $m_A Q_B$ с помощью своего секретного ключа m_A и публичного ключа Q_B Боба. Подобным образом действует и Боб.

Эта система резюмируется в следующей таблице.

СИСТЕМНЫЕ ПАРАМЕТРЫ	ЭЛЛИПТИЧЕСКАЯ КРИВАЯ $\mathcal{E}$ НАД $\mathrm{GF}(q)$ ТОЧКА P НА $\mathcal{E}$ ВЫСОКОГО ПОРЯДКА
СЕКРЕТНЫЙ КЛЮЧ U	m_U
ПУБЛИЧНЫЙ КЛЮЧ U	$Q_U = m_U P$
ОБЩИЙ КЛЮЧ A И B	$K_{A,B} = m_A m_B P$
АЛИСА ВЫЧИСЛЯЕТ	$m_A Q_B$
БОБ ВЫЧИСЛЯЕТ	$m_B Q_A$

Таблица 10.1. Система Диффи–Хеллмана обмена ключами над эллиптическими кривыми.

Пример 10.6

Рассмотрим эллиптическую кривую $\mathcal{E}$ над $\mathbb{Z}_{863}$, заданную уравнением $y^2 = x^3 + 100x^2 + 10x + 1$. Точка $P = (121, 517)$ лежит на ней, что можно проверить посредством функции Mod из пакета "Mathematica":

```
p = 863; a = 100; b = 10; c = 1;
x = 121; y = 517;
Mod[y^2 - (x^3 + a * x^2 + b * x + c), p] == 0
```

```
True
```

Порядок точки P равен 432. Чтобы показать это, проверим, что $432P = O$ и что $(432/q)P \neq O$ для простых делителей q числа 432. Используем двоичные разложения этих коэффициентов (определяемых посредством функции IntegerDigits). Мы используем также функцию EllipticAdd, определенную в разд. 10.3, и функцию Do.

```
FactorInteger[432]
IntegerDigits[432, 2]
IntegerDigits[432 / 2, 2]
IntegerDigits[432 / 3, 2]
```

```
|| {{2, 4}, {3, 3}}

|| {1, 1, 0, 1, 1, 0, 0, 0, 0}

|| {1, 1, 0, 1, 1, 0, 0, 0}

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

```
p = 863; P = .;
a = 100; b = 10; c = 1;
P[0] = {121, 517};
P[i_] := P[i] = EllipticAdd[p, a, b, c, P[i - 1], P[i - 1]];
Q = EllipticAdd[p, a, b, c,EllipticAdd[p, a, b, c,P[8],P[7]],
    EllipticAdd[p, a, b, c, P[5], P[4]]]
EllipticAdd[p, a, b, c, EllipticAdd[p, a, b, c, P[7], P[6]],
    EllipticAdd[p, a, b, c, P[4], P[3]]]
EllipticAdd[p, a, b, c, P[7], P[4]]
```

```
|| {0}

|| {19, 0}

|| {341, 175}
```

Пусть Алиса выбирает $m_A = 130$, *а Боб —* $m_B = 288$. *Тогда* $Q_A = (162, 663)$, *а* $Q_B = (341, 688)$, *что можно проверить следующим образом (заметим, что мы выбрали очень "дружелюбные" секретные скаляры*[2]*):*

```
QAlice = EllipticAdd[p, a, b, c, P[7], P[1]]
QBob = EllipticAdd[p, a, b, c, P[8], P[5]]
```

```
|| {162, 663}

|| {341, 688}
```

Алиса находит общий ключ $K_{A,B}$, *вычисляя* $K_{A,B} = m_A Q_B$, *где* $m_A = 130$ — *ее секретный ключ. Она получает*

```
QA[0] = {341, 688};
QA[i_] := QA[i] =
    EllipticAdd[p, a, b, c, QA[i - 1], QA[i - 1]];
EllipticAdd[p, a, b, c, QA[7], QA[1]]
```

```
|| {341, 688}
```

[2] Весьма "дружелюбен" лишь секретный скаляр Боба. Действительно, поскольку $288 = 2^5 \cdot 3^2$ порядок точки $Q_B = 288 \cdot P$ равен $432/\text{НОД}(288, 432) = 3$ (всего лишь!). Поэтому общий секретный ключ Боба с любым другим участником протокола выбирается из трех точек: $\{0\}$, Q_B и $2 \cdot Q_B = \{341, 175\}$. Секретный скаляр Алисы не так плох, поскольку $432/\text{НОД}(130, 432) = 216$. — *Прим. перев.*

Аналогично, Боб находит общий ключ $K_{A,B}$, *вычисляя* $K_{A,B} = m_B Q_A$, *где* $m_B = 288$ — *его секретный ключ. Он также получает*

```
QB[0] = {162, 663};
QB[i_] := QB[i] =
   EllipticAdd[p, a, b, c, QB[i - 1], QB[i - 1]];
EllipticAdd[p, a, b, c, QB[8], QB[5]]
```

```
{341, 688}
```

Теперь, когда система Диффи–Хеллмана обмена ключами описана, вполне реально выполнить упражнение, показывающее, что протокол Эль-Гамаля и другие системы, описанные в разд. 8.2, могут быть переписаны на языке эллиптических кривых[3].

10.4.3 Безопасность систем дискретных логарифмов, основанных на эллиптических кривых

В разд. 8.3 приведены различные методы взятия дискретных логарифмов над конечным полем. Алгоритм Похлига–Хеллмана, метод "детский шаг– гигантский шаг", ρ-метод Полларда — все они могут быть переведены на язык эллиптических кривых: нужно просто заменить модулярное возведение в степень скалярным умножением на эллиптических кривых.

Во время написания этой книги терпела поражение любая попытка эффективно перевести на язык эллиптических кривых метод исчисления индексов (см. [Mill86]). Это имеет большое криптографическое значение, так как метод исчисления индексов — единственный с субэкспоненциальной сложностью. Это означает, что для обычных систем дискретных логарифмов метод исчисления индексов является господствующим фактором при определении системных параметров (для поддержания вычислительной безопасности систем). Так как метод исчисления индексов не работает в обстановке эллиптических кривых, можно позволить себе много меньшие параметры для достижения того же уровня безопасности.

Во время написания книги был предложен *метод XEDNI* ([Silv98]) как альтернатива для решения проблемы дискретных логарифмов на эллиптических кривых. Но еще требуется дальнейший анализ, чтобы определить последствия внедрения этого метода.

Существуют специальные атаки на криптосистемы дискретных логарифмов, основанные на эллиптических кривых. Из-за этих атак необходимо избегать некоторых классов эллиптических кривых. В частности, **не должны** использоваться

[3]Заметим, что такое "упражнение" было выполнено и в отношении российского стандарта цифровой подписи *[ГОСТП94]: новый стандарт *[ГОСТ01], основанный на эллиптических кривых, в принципе является результатом такого "переписывания". – *Прим. ред.*

- сингулярные кривые;
- суперсингулярные кривые;
- аномальные кривые.

Мы не будем описывать эти атаки (см. [MeOkV93], [SatA98], [Smar98]). В каждом случае проблема логарифмов над эллиптическими кривыми может быть транслирована в проблему логарифмов над конечным полем (или даже в более простую проблему). В одном случае мы объясним, как можно учесть эти атаки, просто избегая соответствующих особых кривых.

Прежде чем сделать это, необходимо ввести новое понятие. Сопоставим уравнению Вейерштрасса (10.1) *однородное* уравнение. А именно, умножим каждый член в (10.1) на наименьшую степень z так, чтобы новые члены имели одну и ту же степень[4]:

$$F(x,y,z) = y^2z + uxyz + vyz^2 - x^3 - ax^2z - bxz^2 - cz^3 = 0. \qquad (10.16)$$

Заметим, что если (x,y,z) удовлетворяет уравнению (10.16), то это же верно для $\lambda(x,y,z)$. По этой причине решения уравнения (10.16) часто нормализуют, требуя, чтобы самая правая ненулевая координата была равна 1. Теперь точки (x,y), удовлетворяющие уравнению (10.1), приводят к решениям $(x,y,1)$ для (10.16). Точка O в бесконечности может быть (несколько таинственно) представлена тройкой $(0,1,0)$.

Точка на кривой $\mathcal{E}$ называется *сингулярной* (или особой), если все частные производные $\partial F/\partial x$, $\partial F/\partial y$ и $\partial F/\partial z$ равны нулю. Эллиптическая кривая не может содержать двух сингулярных точек. Если кривая $\mathcal{E}$ содержит сингулярную точку, то она называется *сингулярной кривой*, в противном случае она называется *несингулярной* (или неособой) *кривой*.

С некоторыми усилиями можно показать, что уравнение (10.2) тогда и только тогда определяет несингулярную кривую, когда кубический многочлен в правой части не имеет кратных корней. Для уравнения (10.3) при $c \neq 0$ это эквивалентно условию $4b^3 + 27c^2 \not\equiv 0 \pmod p$. Когда $p = 2$, несингулярные кривые определяются уравнениями (10.4) при $c \neq 0$ и (10.5) при $v \neq 0$.

Из сказанного следует, что довольно просто проверить, является кривая сингулярной или нет.

Мы не даем определения *суперсингулярных кривых*[5]. Здесь достаточно знать, что кривые, заданные уравнением (10.5), суперсингулярны и нужно избегать их. Это снова легко сделать.

Наконец, *аномальными* называются кривые $\mathcal{E}$ над $\mathbb{Z}_p$ с тем свойством, что $|\mathcal{E}| = p$.

---

[4] Тем самым мы переходим к кривой в проективной плоскости; см. ниже подразд. 13.3.2.— *Прим. ред.*

[5] Согласно теореме 10.1 (Хассе) мощность эллиптической кривой $\mathcal{E}$ над полем $\mathbb{F}_q$ равна $q + 1 - t$, где $|t| \leq 2\sqrt{q}$. Кривая $\mathcal{E}$ называется *суперсингулярной*, если $t^2 = 0, q, 2q, 3q$ или $4q$. — *Прим. ред.*

10.5 Задачи

Задача 10.1^M. Сколько точек лежит на эллиптической кривой, определенной в примере 10.1?

Задача 10.2. Найдите точки пересечения прямых $y = 4x + 20$ и $y = 4x + 21$ с эллиптической кривой $y^2 = x^3 + 25x + 10$ над $\mathbb{Z}_{31}$.

Задача 10.3. Найдите прямую, которая касается эллиптической кривой $y^2 = x^3 + 11x^2 + 17x + 25$ над $\mathbb{Z}_{31}$ в точке $(2, 7)$. Где еще эта прямая пересекает кривую?

Задача 10.4[6]. Рассмотрим эллиптическую кривую $\mathcal{E}$ с уравнением $y^2 = x^3 + 11x^2 + 17x + 25$ над $\mathbb{Z}_{31}$. Проверьте, лежат ли на $\mathcal{E}$ точки $P = (12, 10)$ и $Q = (25, 14)$. Что такое $-P$? Вычислите сумму P и Q без использования процедуры из "Mathematica", представленной в разд. 10.3.

Задача 10.5. Пусть P на $\mathcal{E}$ имеет порядок n. Каков порядок точки $-P$?

Задача 10.6^M. Рассмотрим эллиптическую кривую, определяемую уравнением $y^2 = x^3 + 11x^2 + 17x + 25$ над $\mathbb{Z}_{31}$. Определите порядки точек $P = (27, 10)$ и $Q = (24, 28)$. Какой можно сделать вывод о мощности $\mathcal{E}$ (совет: используйте теорему B.5)? Какова мощность $\mathcal{E}$ (используйте теорему 10.1)? Постройте точку максимального порядка, исходя из точек P и Q.

Задача 10.7^M. Продублируйте пример 10.6 для эллиптической кривой над $\mathbb{Z}_{523}$, определяемой уравнением $y^2 = x^3 + 111x^2 + 11x + 1$. Используйте в качестве P точку порядка не меньше 100.

[6] В оригинале здесь был значок "M". Но задача легко решается вручную.— *Прим. ред.*

Глава 11

Системы, основанные на теории кодирования

11.1 Введение в коды Гоппы

В данной главе предполагается, что читатель знаком с алгебраической теорией кодирования. В противном случае можно свободно пропустить эту главу и приступить к чтению гл. 12. Здесь мы напомним следующие факты из [MacWS77] о кодах Гоппы.

Теорема 11.1.
Пусть $G(x)$ — неприводимый многочлен степени t над полем $GF(2^m)$. Тогда множество

$$\Gamma(G(x), GF(2^m)) = \left\{ (c_\omega)_{\omega \in GF(2^m)} \in \{0,1\}^n \,\middle|\, \sum_{\omega \in GF(2^m)} \frac{c_\omega}{x-\omega} \equiv 0 (\bmod G(x)) \right\} \tag{11.1}$$

определяет двоичный *код Гоппы длины* $n = 2^m$, *размерности* $k \geq n - t \cdot m$, с *минимальным расстоянием* $d \geq 2t+1$. Для него есть быстрый *алгоритм декодирования* со временем работы $n \cdot t$ (см. [Patt75]).

Заметим, что поле $GF(2^m)$ используется в качестве множества индексов для координат векторов из $\{0,1\}^n$. Описанное выше означает, что элементы множества $\Gamma(G(x), GF(2^m))$ (называемые *кодовыми словами*) образуют линейное подпространство в $\{0,1\}^n$ размерности не менее, чем $n - t \cdot m$, и различные кодовые слова отличаются по крайней мере в $2t+1$ координатах (иначе говоря, *расстояние Хэмминга* $d_H(\underline{c}, \underline{c}')$ между различными кодовыми словами $\underline{c}$ и $\underline{c}'$ не меньше, чем $2t+1$).

Алгоритм декодирования будет отображать каждое слово из $\{0,1\}^n$, отличающееся от (единственного в силу неравенства треугольника) кодового слова $\underline{c}$ не более, чем в t координатах, в это кодовое слово. Следовательно, если послано слово $\underline{c}$, а принято слово $\underline{r}$, отличающееся от $\underline{c}$ не более, чем в t координатах ($d_H(\underline{c}, \underline{r}) \leq t$), то получатель может по $\underline{r}$ восстановить $\underline{c}$. Поэтому число t называется *способностью* кода $\Gamma(G(x), GF(2^m))$ *исправлять ошибки*.

Произвольная матрица G размера $k \times n$, строки которой образуют

базис для данного линейного кода, называется *порождающей матрицей* этого кода. Из этого определения следует, что код может быть описан как множество

$$\{\underline{m} \cdot G | \underline{m} \in \{0,1\}^k\}. \tag{11.2}$$

Пример 11.1 (часть 1)

Пусть α — примитивный элемент поля $GF(2^4)$, удовлетворяющий равенству $\alpha^4 + \alpha^3 + 1 = 0$. После загрузки пакета Algebra'FiniteFields', можно с помощью функций MatrixForm и PowerList сгенерировать таблицу логарифмов поля $GF(2^4)$.

```
<< Algebra'FiniteFields'
```

```
MatrixForm[PowerList[GF[2, {1, 0, 0, 1, 1}]]]
```

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

Рассмотрим двоичный код Гоппы $\Gamma(G(x), GF(2^4))$ длины 16, определяемый многочленом $G(x) = x^2 + x + \alpha$. Для проверки того, что $G(x)$ неприводим над $GF(2^4)$, достаточно лишь убедиться в том, что он не имеет линейных множителей, а это легко сделать с помощью функций GF, Table и TableForm пакета "Mathematica".

```
f16 = GF[2, {1, 0, 0, 1, 1}];
x = f16[{0, 1}];
a = f16[{0, 1}];
G[x_] := x^2 + x + a;
G[0]
Table[{i, G[x^i], {i, 0, 14}] // TableForm
```

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

```
0    {0, 1, 0, 0}2
1    {0, 0, 1, 0}2
2    {1, 1, 1, 1}2
3    {1, 0, 1, 0}2
4    {1, 0, 1, 0}2
5    {1, 1, 0, 0}2
6    {0, 1, 1, 1}2
7    {1, 0, 0, 1}2
8    {0, 1, 1, 1}2
9    {1, 1, 1, 1}2
10   {1, 1, 0, 0}2
11   {0, 0, 0, 1}2
12   {0, 0, 1, 0}2
13   {1, 0, 0, 1}2
14   {0, 0, 0, 1}2
```

Для нахождения обратных элементов $1/(x-\alpha) \pmod{x^2+x+\alpha}$ *в формуле* (11.1) *используем пакет Algebra'PolynomialExtendedGCD'*

```
<< Algebra'PolynomialExtendedGCD'
```

и функцию PolynomialExtendedGCD пакета "Mathematica". Например, $1/(x-\alpha^3) \pmod{x^2+x+\alpha}$ *можно найти так:*

```
x = .;
PolynomialExtendedGCD[x - a^3, x^2 + x + a]
```

```
{1, {{0, 1, 0, 1}2 + x{1, 1, 1, 1}2, {1, 1, 1, 1}2 }}
```

С помощью таблицы логарифмов можно переписать эти коэффициенты следующим образом:

$$0 \cdot 1 + 1 \cdot \alpha + 0 \cdot \alpha^2 + 1 \cdot \alpha^3 = \alpha^{10},$$

$$1 \cdot 1 + 1 \cdot \alpha + 1 \cdot \alpha^2 + 1 \cdot \alpha^3 = \alpha^6.$$

Из (A.8) *следует, что*

$$(x-\alpha^3) \cdot (\alpha^{10} + \alpha^6 x) + \alpha^6 \cdot G(x) = 1,$$

т.е. $1/(x-\alpha^3) = \alpha^{10} + \alpha^6 x$. *Это можно проверить с помощью функции PolynomialMod пакета "Mathematica".*

```
Clear[x];
PolynomialMod[(x-a^3) * (a^10 + a^6x), x^2 + x + a]
```

```
{1, 0, 0, 0}2
```

Таким способом выразим все обратные $1/(x-\omega)$, $\omega \in GF(2^4)$, *в виде многочленов* $g_0^{(\omega)} + g_1^{(\omega)} x$.

```
Clear[x];
PolynomialExtendedGCD[x, x^2 + x + a]
Do[Print[PolynomialExtendedGCD[x - a^i, x^2 + x + a]],
  {i, 0, 14}]
```

```
{1, {{0, 0, 1, 1}2 + x{0, 0, 1, 1}2, {0, 0, 1, 1}2 }}
{1, {x{0, 0, 1, 1}2, {0, 0, 1, 1}2 }}
{1, {{0, 1, 0, 1}2 + x{0, 1, 1, 0}2, {0, 1, 1, 0}2 }}
{1, {{0, 0, 0, 1}2 + x{1, 0, 1, 0}2, {1, 0, 1, 0}2 }}
{1, {{0, 1, 0, 1}2 + x{1, 1, 1, 1}2, {1, 1, 1, 1}2 }}
{1, {{1, 0, 1, 0}2 + x{1, 1, 1, 1}2, {1, 1, 1, 1}2 }}
{1, {{0, 1, 1, 0}2 + x{0, 0, 0, 1}2, {0, 0, 0, 1}2 }}
{1, {{1, 0, 0, 0}2 + x{1, 1, 1, 0}2, {1, 1, 1, 0}2 }}
{1, {{1, 0, 1, 0}2 + x{1, 0, 1, 1}2, {1, 0, 1, 1}2 }}
{1, {{0, 1, 1, 0}2 + x{1, 1, 1, 0}2, {1, 1, 1, 0}2 }}
{1, {{1, 0, 1, 1}2 + x{1, 0, 1, 0}2, {1, 0, 1, 0}2 }}
{1, {{0, 1, 1, 1}2 + x{0, 0, 0, 1}2, {0, 0, 0, 1}2 }}
{1, {{1, 0, 1, 1}2 + x{1, 1, 0, 0}2, {1, 1, 0, 0}2 }}
{1, {{0, 0, 1, 1}2 + x{0, 1, 1, 0}2, {0, 1, 1, 0}2 }}
{1, {{0, 0, 0, 1}2 + x{1, 0, 1, 1}2, {1, 0, 1, 1}2 }}
{1, {{0, 1, 1, 1}2 + x{1, 1, 0, 0}2, {1, 1, 0, 0}2 }}
```

Затем представим их в виде столбцов $\begin{pmatrix} g_0^{(\omega)} \\ g_1^{(\omega)} \end{pmatrix}$ *матрицы* H *размером* 2×16. *Отметим, что столбец* $\begin{pmatrix} \alpha^{10} \\ \alpha^6 \end{pmatrix}$, *соответствующий* $1/(x-\alpha^3)$, *располагается в* 5*-м столбце матрицы* H, *поскольку первый столбец соответствует значению* $\omega = 0$, *второй — значению* $\omega = 1$ *и т.д.*

$$H = \begin{pmatrix} \alpha^{14} & 0 & \alpha^{10} & \alpha^3 & \alpha^{10} & \alpha^9 & \alpha^{13} & 1 & \alpha^9 & \alpha^{13} & \alpha^{11} & \alpha^8 & \alpha^{11} & \alpha^{14} & \alpha^3 & \alpha^8 \\ \alpha^{14} & \alpha^{14} & \alpha^{13} & \alpha^9 & \alpha^6 & \alpha^6 & \alpha^3 & \alpha^7 & \alpha^{11} & \alpha^7 & \alpha^9 & \alpha^3 & \alpha^{12} & \alpha^{13} & \alpha^{11} & \alpha^{12} \end{pmatrix}$$

Теперь применим вычисленную ранее таблицу логарифмов поля $GF(2^4)$. *Определяющее соотношение в* (11.1) *можно переписать в виде*

$$\sum\nolimits_{\omega \in GF(2^4)} c_\omega (g_0^{(\omega)} + g_1^{(\omega)} x) \equiv 0 \pmod{x^2 + x + \alpha},$$

или, что эквивалентно,

$$\sum\nolimits_{\omega \in GF(2^4)} c_\omega g_0^{(\omega)} + (\sum\nolimits_{\omega \in GF(2^4)} c_\omega g_1^{(\omega)}) x \equiv 0 \pmod{x^2 + x + \alpha}.$$

Таким образом, получаем два линейных уравнения для $\underline{c} = (c_\omega)_{\omega \in GF(2^4)}$:

$$\sum\nolimits_{\omega \in GF(2^4)} c_\omega g_0^{(\omega)} = 0 \text{ и } \sum\nolimits_{\omega \in GF(2^4)} c_\omega g_1^{(\omega)} = 0.$$

Эту пару уравнений можно эффективно записать в виде

$$H \cdot \underline{c}^T = \underline{0}^T.$$

Представление каждой степени α *в виде двоичной линейной комбинации* 1, α, α^2 *и* α^3 *(или использование результатов работы функции* PolynomialExtendedGCD *напрямую) дает двоичную матрицу* H' *размером* 8×16:

$$H' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Таким образом, другим способом описания $\Gamma(x^2 + x + \alpha, GF(2^4))$ *является*

$$C = \{\underline{c} \in \{0,1\}^{16} | H' \cdot \underline{c}^T = \underline{0}^T\}.$$

Нетрудно убедиться, что C *— это двоичный линейный код длины* 16, *размерности* 8 *с минимальным расстоянием* 5.

Матрица H, нуль-пространством которой является данный код C, называется *проверочной матрицей* кода C. Можно записать

$$C = \{\underline{c} \in \{0,1\}^n | H' \cdot \underline{c}^T = \underline{0}^T\}. \tag{11.3}$$

Синдром полученного вектора $\underline{r}$ определяется как $\underline{s}^T = H \cdot \underline{r}^T$.

Количество неприводимых многочленов степени t над полем $GF(2^m)$ приблизительно равно $2^{m \cdot t}/t$ (см. следствие B.18). Таким образом, наугад выбранный многочлен степени t над полем $GF(2^m)$ будет неприводимым с вероятностью $1/t$. Поскольку существуют быстрые алгоритмы проверки неприводимости (см. [Berl68], гл. 6, или [Rabi80]), можно найти неприводимый многочлен степени t над полем $GF(2^m)$ так же, как и в алг. 9.3, выбирая случайный многочлен и проверяя неприводимость.

11.2 Криптосистема Мак-Элиса

Основываясь на теории кодов, исправляющих ошибки, Мак-Элис ([McEl78]) предложил следующую систему шифрования[1].

11.2.1 Система

□ Формирование системы

1) Каждый пользователь U выбирает подходящий код Гоппы длины $n_U = 2^{m_U}$, исправляющий t_U ошибок. Для этого пользователь U выбирает случайный неприводимый многочлен $p_U(x)$ степени t_U над полем $GF(2^{m_U})$ и находит порождающую матрицу G_U соответствующего кода Гоппы $\Gamma(p_U(x), GF(2^{m_U}))$. Это матрица размера $k_U \times n_U$.

2) Пользователь U выбирает случайную невырожденную матрицу S_U размером $k_U \times k_U$ и случайную матрицу перестановки[2] P_U размером $n_U \times n_U$, а затем вычисляет

$$G_U^* = S_U G_U P_U. \tag{11.4}$$

3) Пользователь U объявляет G_U^* и t_U публичными, а G_U, S_U и P_U хранит в секрете.

□ Шифрование

Предположим, что пользователь Алиса хочет послать сообщение Бобу. Она находит параметры G_B^* (размером $k_B \times n_B$) и t_B, объявленные Бобом публичными, и представляет свое сообщение в виде двоичной строки $\underline{m}$ длины k_B. Затем Алиса выбирает случайный вектор $\underline{e}$ (модель ошибки) длиной n_B, в котором не более, чем t_B, координат равны 1. В качестве полученного из $\underline{m}$ шифртекста Алиса посылает Бобу

$$\underline{c} = \underline{m} \cdot G_B^* + \underline{e}. \tag{11.5}$$

(Обычно говорят, что вес $\underline{e}$ не больше, чем t_B, обозначая его через $w_H(\underline{e}) \leq t_B$, где значение *весовой* функции w равно количеству ненулевых координат вектора.)

□ Дешифрование

После получения $\underline{c}$ Боб с помощью секретной матрицы перестановки P_B вычисляет

$$\underline{c} \cdot P_B^{-1} \overset{(11.5)}{=} \underline{m} \cdot G_B^* P_B^{-1} + \underline{e} \cdot P_B^{-1} \overset{(11.4)}{=} \underline{m} \cdot S_B G_B P_B P_B^{-1} + \underline{e}' = (\underline{m} \cdot S_B) G_B + \underline{e}',$$

[1] На русском языке описание этой системы можно найти также в *[Слоэ83].— *Прим. ред.*

[2] Т.е. матрицу, у которой в каждом столбце и каждой строке — ровно по одной единице, а остальные элементы равны 0. — *Прим. перев.*

где $\underline{e}' = \underline{e} \cdot P_B^{-1}$ — перестановка вектора $\underline{e}$, а, значит, имеет вес не более t_B. С помощью алгоритма декодирования для кода Гоппы $\Gamma(p_U(x), GF(2^{m_U}))$ Боб может эффективно декодировать $\underline{c} \cdot P_B^{-1}$. Он найдет $\underline{e}'$ как ошибку и сможет восстановить $\underline{m} \cdot S_B$. Умножение этого выражения справа на матрицу S_B^{-1} (известную Бобу) дает первоначально пересылаемое сообщение $\underline{m} \in \{0,1\}^{k_B}$.

11.2.2 Обсуждение

☐ Резюме и предполагаемые параметры

Введенную в предыдущем разделе криптосистему можно резюмировать в следующей таблице.

ПУБЛИЧНЫЕ	G_U^* И t_U ВСЕХ ПОЛЬЗОВАТЕЛЕЙ U, G_U^* ИМЕЕТ РАЗМЕРЫ $k_U \times n_U$
СЕКРЕТНЫЕ	$p_U(x)$, S_U И P_U КАЖДОГО ПОЛЬЗОВАТЕЛЯ U
СВОЙСТВО	$S_U^{-1} G_U^* P_U$ — ПОРОЖДАЮЩАЯ МАТРИЦА КОДА ГОППЫ, ОПРЕДЕЛЯЕМОГО МНОГОЧЛЕНОМ $p_U(x)$ СТЕПЕНИ t_U
ФОРМАТ СООБЩЕНИЯ АЛИСЫ БОБУ	$\underline{m} \in \{0,1\}^{k_B}$
ШИФРОВАНИЕ	$\underline{c} = \underline{m} \cdot G_B^* + \underline{e}$ ВЕС $\underline{e}$ НЕ БОЛЕЕ t_B
ДЕШИФРОВАНИЕ	ВЫЧИСЛИТЬ $\underline{c}' = \underline{c} \cdot P_B^{-1}$ ДЕКОДИРОВАТЬ $\underline{c}'$, ЧТОБЫ НАЙТИ $\underline{m}' = \underline{m} \cdot S_B$ ВЫЧИСЛИТЬ $\underline{m}' \cdot S_B^{-1} = \underline{m}$

Таблица 11.1. Криптосистема Мак-Элиса.

Смысл вектора ошибки $\underline{e}$, введенного в формуле (11.5), разумеется, в том, чтобы лишить криптоаналитика возможности использовать при восстановлении $\underline{m}$ по $\underline{c}$ исключение неизвестных методом Гаусса.

Мак-Элис первоначально ([McEl78]) предложил брать $m_B = 10$ (тогда $n_B = 1024$) и $t_B = 50$ ($k_B \approx 1024 - 50 \times 10 = 524$).

☐ Эвристика схемы

Нетрудно угадать эвристику, стоящую за этой схемой. Возьмем достаточно длинный двоичный линейный блочный код, который способен исправлять большое число, скажем, t, ошибок, и для которого существует

эффективный алгоритм декодирования. Если код выбирается из достаточно большого класса кодов, то угадать, какой именно код был выбран, практически невозможно. Пусть n — длина кода, а k — его размерность. Преобразуем порождающую матрицу до такой степени, чтобы получившаяся в результате матрица выглядела как случайная $k \times n$ матрица ранга k. Сложность декодирования случайно порожденного кода с такими параметрами должна быть высокой. Сложность некоторых алгоритмов декодирования будет обсуждена в следующем разделе. В [BerMT77] показано, что общая задача декодирования линейных кодов, т.е. задача поиска ближайшего кодового слова для произвольного слова длины n, является *NP-полной*. Мы не будем здесь объяснять, что конкретно означает это понятие. Заинтересованный читатель может обратиться к [GarJ79].

Здесь нам будет достаточно знать об этой характеризации лишь то, что не известно ни одного алгоритма, который находил бы по произвольному слову ближайшее к нему кодовое слово и работал за время, полиномиально зависящее от длины входа.

Более того, если такой алгоритм удастся найти, то его можно будет адаптировать для решения большого класса[3] эквивалентных по сложности задач.

□ Отсутствие режима подписывания

Функция шифрования в криптосистеме Мак-Элиса отображает двоичные последовательности длины k в двоичные последовательности длины n. Это отображение не является сюръективным. Действительно, при предполагаемых параметрах количество векторов длины n, расположенных на расстоянии ≤ 50 от кодовых слов, равно

$$2^k \sum_{i=0}^{50} \binom{n}{i} \approx 2^{524} \sum_{i=0}^{50} \binom{1024}{i} \approx 2^{808.4},$$

что составляет весьма незначительную долю от всех слов длины 1024. Таким образом, (секретная) функция S_U, упомянутая в свойстве ПК4 из разд. 7.1.1, не определена на большинстве слов из $\{0,1\}^n$. Следовательно, криптосистему Мак-Элиса нельзя использовать в режиме подписывания сообщений. См. также табл. 7.2.

11.2.3 Аспекты безопасности

Теперь обсудим безопасность криптосистемы Мак-Элиса, анализируя четыре возможных способа атаки, при условии использования значений параметров, предложенных Мак-Элисом. (Большинство известных к настоящему моменту сильных атак есть в [CanS98].)

```
n = 1024; k = 524; t = 50;
```

[3] Вообще-то, всех.— *Прим. ред.*

☐ Угадывание S_B и P_B

Криптоаналитик может попытаться угадать S_B и P_B для того, чтобы найти G_B по G_B^*, используя равенство (11.4). Когда матрица G_B станет известна криптоаналитику, ему не составит труда найти многочлен $p_U(x)$, определяющий код Гоппы $\Gamma(p_U(x), GF(2^{m_U}))$ с порождающей матрицей G_B. Затем, действуя по алгоритму дешифрования Боба, он сможет определить посланное сообщение $\underline{m}$.

Однако количество обратимых матриц S_B и матриц перестановок P_B измеряется астрономическими числами ($\Pi_{i=0}^{k-1}(2^k - 2^i)$ и, соответственно, $n!$). Из-за этого вероятность успеха при такой атаке меньше, чем при попытке угадать вектор $\underline{m}$ напрямую.

☐ Сравнение со всеми кодовыми словами

Криптоаналитик может сравнить полученный вектор $\underline{r}$ со всеми кодовыми словами кода, порожденного G_B^*. Пусть $\underline{c}$ — ближайшее кодовое слово. Оно находится на расстоянии $\leq t$ от $\underline{r}$ (по правилу шифрования (11.5)). Такое слово единственно, поскольку минимальное расстояние кода не менее $2t+1$. Кроме того, из формулы (11.5) следует, что $\underline{c} = \underline{m} \cdot G_B^*$. Исключая неизвестные методом Гаусса, из $\underline{c}$ можно восстановить $\underline{m}$.

Этот подход требует следующего числа сравнений.

```
N[2^k, 5]
```

```
5.4918×10^157
```

Пример 11.2 (часть 1)

Рассмотрим двоичный код длины $n = 7$ и размерности $k = 4$, порожденный матрицей

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix};$$

и предположим, что $\underline{r} = (1,1,1,0,1,0,1)$ — перехваченный шифртекст, представляющий собой сумму кодового слова $\underline{c}$ и вектора ошибок с весом не более, чем 1 (так как $t = 1$).

Используя функцию Mod, сравним $\underline{r}$ с двумя (вместо $2^k = 16$) кодовыми словами:

```
r = {1, 1, 1, 0, 1, 0, 1};
i1 = {1, 1, 1, 1};
c = Mod[i1 * G, 2]
Mod[r - c, 2]
```

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

Таким образом, $\underline{c} = \underline{i}_1 \cdot G$ *лежит на расстоянии* ≥ 2 *от* $\underline{r}$*, а это слишком далеко.*

```
i2 = {1, 0, 1, 0};
c = Mod[i2 * G, 2]
Mod[r - c, 2]
```

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

Теперь $\underline{c} = \underline{i}_2 \cdot G$ *лежит на расстоянии* 1 *от* $\underline{r}$*, и можно заключить, что был передан вектор* $(1,0,1,0)$.

□ Декодирование по синдрому

Криптоаналитик может вычислить проверочную матрицу H_B^* из уравнения $H_B^* \cdot G_B^* = O$ (см. (11.3)). Ее ранг равен $n-k$. Затем можно породить все векторы ошибок $\underline{e}$ с весом не более, чем t, вычислить для каждого из них синдром $H_B^* \cdot \underline{e}^T$ и поместить их в таблицу.

Для перехваченного вектора $\underline{r}$ сперва вычисляется синдром $s^T = H_B^* \cdot \underline{r}^T$. Затем по таблице можно найти соответствующий вектор ошибок $\underline{e}$. Вычитая $\underline{e}$ из $\underline{r}$, можно получить кодовое слово $\underline{c} = \underline{m} \cdot G_B^*$ (см. (11.5)). Исключая неизвестные методом Гаусса, из $\underline{c}$ можно восстановить переданное сообщение $\underline{m}$.

Объем работы, которую необходимо выполнить при проведении этой атаки — $\sum_{i=0}^{50} \binom{n}{i}$ (столько строк будет в построенной таблице):

```
N[Sum[Binomial[n, i], {i, 0, 50}], 5]
```

$$3.3623\times10^{85}$$

Пример 11.2 (часть 2)

Проверочная матрица кода, введенного в примере 11.2*, задается так:*

```
H = {{1, 1, 0, 1, 1, 0, 0},
     {1, 0, 1, 1, 0, 1, 0},
     {0, 1, 1, 1, 0, 0, 1}};
MatrixForm[H]
```

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

Это можно проверить с помощью функций Transpose и MatrixForm пакета "Mathematica":

```
U = Mod[G * Traspose[H], 2];
MatrixForm[H]
```

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

Затем породим все векторы ошибки $\underline{e}$ весом ≤ 1 и вычислим их синдромы $H_B^* \cdot \underline{e}^T$. Поместим все это в таблицу. Кроме функций Mod, Do и Print пакета "Mathematica", применим функцию ReplacePart, которая заменяет i-ю компоненту вектора $\underline{e}$ заданной величиной (здесь – ее дополнением).

```
e = {0, 0, 0, 0, 0, 0, 0};
Print[e, " ", Mod[H * e, 2]];
Do[ {er = ReplacePart[e, Mod[e[i]] + 1, 2], i],
     Print[er, " ", Mod[H * er, 2]]}, {i, 1, 7}]
```

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

С помощью данной таблицы легко найти кодовое слово на расстоянии ≤ 1 от $\underline{r}$.

```
r = {1, 1, 1, 0, 1, 0, 1};
Mod[H * r, 2]
```

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

Это синдром, соответствующий вектору $\underline{e} = \{1,0,0,0,0,0,0\}$, значит, ближайшее кодовое слово определяется как

```
e = {0, 1, 0, 0, 0, 0, 0};
Mod[r - e, 2]
```

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

Поскольку порождающая матрица G в этом примере имеет вид $(I_4|P)$, можно извлечь переданную информацию $\underline{m}$ из четырех первых компонент вектора $\underline{c}$:

$$\underline{m} = \{1, 0, 1, 0\}.$$

□ Угадывание k правильных и независимых компонент

Криптоаналитик наугад выбирает k позиций и надеется, что в них нет ошибки, т.е. он надеется на то, что в этих компонентах вектора $\underline{e}$ оказались нули. Если у сужения матрицы G_B^* на эти k позиций по-прежнему ранг равен k, то можно найти кандидата $\underline{m}'$ на роль информационного вектора $\underline{m}$ методом Гаусса.

Если ранг меньше k, то высока вероятность того, что он близок к k (см. задачу 11.2). Тогда метод Гаусса приведет либо к небольшому числу вариантов для $\underline{m}'$, либо вообще к отсутствию решения.

Для каждого возможного кандидата $\underline{m}'$ вычисляется $\underline{m}' \cdot G_B^*$ и проверяется, лежит ли этот вектор на расстоянии $\leq t$ от перехваченного вектора $\underline{r}$. Если это так, то найден правильный вектор $\underline{m}$.

Вероятность того, что k позиций правильны, — около $(1 - t/n)^k$. Алгоритм Гаусса выполняется за k^3 шагов. Таким образом, средняя сложность этого метода атаки

```
N[k^3 * (1 - t/n)^-k, 5]
```

```
3.5504×10^19
```

Хотя эта атака и наиболее эффективная среди всех, она все равно неосуществима.

Пример 11.2 (часть 3)

Угадав, что компоненты 2, 4, 5 и 7 свободны от ошибок в примере 11.2, применим функции Transpose и MatrixForm пакета "Mathematica" для того, чтобы получить сужение $\overline{G'}$ порождающей матрицы G на выбранные компоненты и сужение $\underline{r}'$ перехваченного вектора $\underline{r}$ из примера 11.2 на выбранные компоненты.

```
G = ( 1 0 0 0 1 1 0
      0 1 0 0 1 0 1
      0 0 1 0 0 1 1
      0 0 0 1 1 1 1 );
Guess = {2, 4, 5, 7}
RestrG = Transpose[G] * [[Guess]];
MatrixForm[Transpose[RestrG]]
```

```
{2, 4, 5, 7}
```

```
|| ( 0 0 1 0 )
   ( 1 0 1 1 )
   ( 0 0 0 1 )
   ( 0 1 1 1 )
```

```
r = {1, 1, 1, 0, 1, 0, 1};
rRestr = r[[Guess]]
```

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

Применим функции LinearSolve, NullSpace *и* Transpose *пакета "Mathematica", чтобы увидеть, имеет ли уравнение решение.*

```
LinearSolve[RestrG, rRestr, Modulus -> 2]
NullSpace[RestrG, Modulus -> 2]
```

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

```
|| {}
```

Ясно, что сужение матрицы G не вырождено. Решение $(0,1,0,0)$ *приводит к кодовому слову на расстоянии* ≥ 2 *от* r.

```
m1 = {0, 1, 0, 0};
Mod[r - m1 * G, 2]
```

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

Попробуем угадать еще раз.

```
Guess = {1, 3, 6, 7}
RestrG = Transpose[G] [[Guess]];
MatrixForm[Transpose[RestrG]]
```

```
|| ( 1 0 1 0 )
   ( 0 0 0 1 )
   ( 0 1 1 1 )
   ( 0 0 1 1 )
```

```
r = {1, 1, 1, 0, 1, 0, 1};
rRestr = r[[Guess]]
```

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

```
LinearSolve[RestrG, rRestr, Modulus -> 2]
NullSpace[RestrG, Modulus -> 2]
```

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

```
|| {}
```

Теперь решение $(1,0,1,0)$ *приводит к кодовому слову на расстоянии* ≤ 1 *от* $\underline{r}$.

```
m = {1, 0, 1, 0};
Mod[r - m * G, 2]
```

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

Можно заключить, что был передан вектор $(1,0,1,0)$.

Чтобы реализовать угадывание в "Mathematica", необходимо загрузить пакет *DiscreteMath'Combinatorica'*

```
<< DiscreteMath'Combinatorica'
```

после этого возможно использование функции *RandomKSubset* пакета "Mathematica".

```
RandomKSubset[{1, 2, 3, 4, 5, 6, 7}, 4]
```

```
{2, 3, 4, 6}
```

□ Многократные шифрования одного сообщения

Небезопасно шифровать одно и то же сообщение несколько раз с помощью одной и той же матрицы G_B^*. Чтобы увидеть это, рассмотрим два шифрования сообщения $\underline{m}$, скажем $\underline{r} = \underline{m} \cdot G_B^* + \underline{e}$ и $\underline{r}' = \underline{m} \cdot G_B^* + \underline{e}'$ (см.(11.5)). Можно быть уверенным, что в тех позициях, где $\underline{r}$ и $\underline{r}'$ не совпадают, либо в $\underline{e}$, либо в $\underline{e}'$ стоят 1. И можно быть почти уверенным, что те позиции, в которых $\underline{r}$ и $\underline{r}'$ совпадают, не содержат ошибок.

Более точно, если векторы ошибки $\underline{e}$ и $\underline{e}'$ выбраны случайно, как это и должно быть, то в среднем возникают следующие значения

$(\underline{e}_i, \underline{e}'_i)$	число компонент
(0,0)	$(n-t)^2/n$
(0,1) или (1,0)	$2t(n-t)/n$
(1,1)	t^2/n

Например, при выборе параметров $n = 1024$ и $t = 50$ получаем, что $\underline{e} = \underline{e}' = 1$ в среднем в $50^2/1024 \approx 2.44$ компонентах.

Также в среднем в

```
n = 1024; t = 50;
N[((n - t)^2 + t^2)/n, 3]
```

```
929
```

компонентах $\underline{r}$ и $\underline{r}'$ совпадают. Вероятно, что не более трех из них искажены.

Удаляя любые t^2/n элементов из множества компонент, в которых $\underline{r}$ и $\underline{r}'$ совпадают, можно с высокой степенью уверенности считать, что на

оставшемся множестве компонент ошибок нет, и сужение матрицы G_B^* на это множество имеет ранг k (см. задачу 11.2). Тогда восстановление $\underline{m}$ по $\underline{r}$ производится методом Гаусса.

Когда одно сообщение шифруется более двух раз, взлом системы еще более облегчается.

11.2.4 Маленький пример системы Мак-Элиса

Пример 11.1 (часть 2)

У кода Гоппы $\Gamma(x^2 + x + \alpha, GF(2^4))$ *из примера* 11.1 *порождающая матрица* G *может быть вычислена, исходя из проверочной матрицы, с использованием функции* NullSpace *пакета "Mathematica".*

```
H = ( 0 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0
      0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 1
      1 0 0 0 0 1 1 0 1 1 1 1 1 1 0 1
      1 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1
      0 0 0 1 1 1 0 1 1 1 1 0 1 0 1 1
      0 0 1 0 1 1 0 1 0 1 0 0 1 1 0 1
      1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 0
      1 1 0 0 1 1 1 0 1 0 0 1 0 0 1 0 );
G = NullSpace[H, Modulus -> 2];
MatrixForm[G]
```

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

Порождающую матрицу G *кода* $\Gamma(x^2+x+\alpha, GF(2^4))$ *можно следующим образом превратить в* $G^* = S \cdot G \cdot P$, *где* S *— обратимая матрица, а* P *— матрица перестановки.*

```
S = ( 1 1 1 1 1 1 1 1
      0 1 1 1 1 1 1 1
      0 0 1 1 1 1 1 1
      0 0 0 1 1 1 1 1
      0 0 0 0 1 1 1 1
      0 0 0 0 0 1 1 1
      0 0 0 0 0 0 1 1
      0 0 0 0 0 0 0 1 );
```

```
P = {{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0}};
Gstar = Mod[S * G * P, 2];
MatrixForm[Gstar]
```

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

Кодовое слово, соответствующее информационной последовательности $(1,1,0,0,1,0,0,1)$, *имеет вид*

```
m = {1, 1, 0, 0, 1, 0, 0, 1};
err = {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0};
cw = Mod[m * Gstar + err, 2]
```

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

Заметим, что ошибки были сделаны в 5*-й и* 9*-й компонентах.*

У перехватчика нет эффективного алгоритма для нахождения по слову $\underline{cw}$ *информационного вектора* $\underline{m}$.

Легальный получатель сперва вычисляет $\underline{cd} = \underline{cw} \cdot P^{-1}$ *с помощью функции* Inverse *пакета "Mathematica".*

```
PInv = Inverse[P, Modulus -> 2];
cd = cw * PInv
```

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

Затем этот вектор нужно декодировать с помощью алгоритма декодирования кода Гоппы $\Gamma(x^2+x+\alpha, GF(2^4))$. *Этот алгоритм здесь не обсуждается. В результате получается вектор* $\underline{m}' = (1,0,0,0,1,1,1,0)$. *Это можно проверить, вычислив* $\underline{m}' \cdot G$ *и сравнив с* $\underline{cd}$. *Разность* $\underline{err}'$ *этих векторов имеет вес 2 и совпадает с* $\underline{err} \cdot P^{-1}$.

```
mpr = {1, 0, 0, 0, 1, 1, 1, 0};
errpr = Mod[mpr * G - cd, 2]
err * PInv
```

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

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

Чтобы найти $\underline{m}$, *легальный получатель вычисляет* $\underline{m}' \cdot S^{-1}$.

```
mpr = {1, 0, 0, 0, 1, 1, 1, 0};
InvS = Inverse[S, Modulus -> 2];
Mod[mpr * InvS, 2]
```

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

Это и в самом деле исходное сообщение.

11.3 Другая техника декодирования линейных кодов

В прошлом было приложено много усилий в поиске алгоритмов декодирования для произвольных линейных кодов. Криптосистема Мак-Элиса только интенсифицировала эти поиски. Большинство из этих алгоритмов относятся к тому типу, который выше уже обсуждался: найти k компонент, на которых сужение порождающей матрицы сохраняет ранг и полученный вектор не содержит ошибок. Такая техника называется *декодированием по информационному множеству.*

Здесь мы опишем технику, введенную ван Тилбургом [vTbu88] (см. также [LeeB88]).

Алгоритм 11.2 (*техника обмена битов местами*)
Пусть G — порождающая матрица двоичного кода C длины n, размерности k, с минимальным расстоянием d.
Пусть $\underline{r} = \underline{c} + \underline{e}$ — полученный вектор, $\underline{c} \in C$ (скажем, $\underline{c} = \underline{m} \cdot G$), вес вектора $\underline{e}$ не более t, а $2t+1 \leq d$.

Шаг 1: Применим к G подходящее элементарное преобразование по строкам и подходящую перестановку столбцов, чтобы привести G к так называемому стандартному виду, т.е.

$S \cdot G \cdot P = (I_k|A)$. Положим $\underline{r}' = \underline{r} \cdot P$ и запишем $\underline{r}' = (\underline{r}'_1, \underline{r}'_2)$, где длина $\underline{r}'_1$ равна k. Заметим, что $\underline{r} = \underline{m} \cdot G \cdot P + \underline{e} \cdot P = \underline{m} \cdot S^{-1} \cdot (I_k|A) + \underline{e}'$, где $\underline{e}$ и $\underline{e}'$ — одного веса.

Шаг 2: Положим $\underline{c}' = \underline{r}'_1 \cdot (I_k|A)$. Первые k компонент векторов $\underline{c}'$ и $\underline{r}'$ одинаковы.

Шаг 3: Если $\underline{c}'$ и $\underline{r}'$ различаются не более, чем в t компонентах, то заключаем, что в первых t компонентах ошибок нет. Вычисляем $\underline{m}$ из равенства $\underline{r}' = \underline{m} \cdot S^{-1}$ методом Гаусса и завершаем работу.

Шаг 4: Если $\underline{c}'$ и $\underline{r}'$ различаются более, чем в t компонентах, то выбираем случайный номер строки i, $1 \le i \le k$, и случайный номер столбца j, $1 \le j \le n - k$, так, что $A_{i,j} \neq 0$. Строим из $(I_k|A)$ новую матрицу G, меняя местами i-й и $(k+j)$-й столбцы (i-й столбец в I_k меняется местами с j-м столбцом в A). Возвращаемся к шагу 1, но теперь при приведении матрицы к стандартному виду используем лишь элементарные преобразования строк с помощью i-й строки.

Продемонстрируем один раунд работы приведенного выше алгоритма. Продолжим пример 11.2.

Пример 11.2 (часть 4)

```
G = ( 1 0 0 0 1 1 0 )
    ( 0 1 0 0 1 0 1 )
    ( 0 0 1 0 0 1 1 )
    ( 0 0 0 1 1 1 1 );
MatrixForm[G]
r = {1, 1, 1, 0, 1, 0, 1}
```

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

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

Матрица G уже имеет стандартный вид. Также видно, что кодовое слово $\underline{c}'$, построенное по четырем первым компонентам $\underline{r}$, находится на расстоянии 2 от $\underline{r}$.

```
r1 = Take[r, 4]
cc = Mod[r1 * G, 2]
Mod[r - cc, 2]
```

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

Для обмена местами мы наугад выберем из столбцов $5-7$ *матрицы* G *ненулевой элемент. Пусть это* $G_{2,5}$. *Поменяем местами* 2-*й и* 5-*й столбцы матрицы* G, *используя функцию:*

```
ColumnSwap[B_, i_, j_] := Module[{U, V}, U = Transpose[B];
  V = U[[i]]; U[[i]] = U[[j]]; U[[j]] = V; Transpose[U]]
```

```
G1 = ColumnSwap[G, 2, 5];
MatrixForm[G1]
```

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

Приведем эту матрицу к стандартному виду с помощью функции RowReduce *пакета "Mathematica".*

```
G2 = RowReduce[G1, Modulus -> 2];
MatrixForm[G2]
```

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

Для того, чтобы проанализировать сложность алгоритма побитового обмена, обозначим через $\Pr(l+u|l)$ вероятность того, что ровно $l+u$ из первых k позиций вектора $\underline{e}$ содержат ошибки после перестановки при условии, что до перестановки таких позиций было ровно l ($u=-1,0,1$).

Пусть $a=\min\{t,k\}$. Тогда выполняются следующие очевидные соотношения:

$$\Pr(l-1|l)=\frac{l}{k}\times\frac{n-k-t+l}{n-k}, \quad \text{если } 1\le l\le a, \tag{11.6}$$

$$\Pr(l+1|l)=\frac{k-l}{k}\times\frac{t-l}{n-k}, \quad \text{если } 1\le l\le a-1, \tag{11.7}$$

$$\Pr(l|l)=\begin{cases} 1-\Pr(l-1|l)-\Pr(l+1|l), & \text{если } 1\le l\le a-1,\\ 1-\Pr(l-1|l), & \text{если } l=a. \end{cases} \tag{11.8}$$

Пример 11.3 (часть 1)

Рассмотрим (двоичный) код с параметрами $n=23$, $k=12$ *и* $t=3$. *Тогда* $a=\min\{k,t\}=3$. *Значения* $Pr(l-1|l)$ *и* $Pr(l+1|l)$ *можно вычислить (и напечатать) по формулам* (11.6) *и* (11.7) *с помощью функций* Min, Do *и* Print *пакета "Mathematica".*

```
n = 23; k = 12; t = 3;
a = Min[k, t];
PrDoun[l_] := l * (n - k - t + l) / (k * (n - k));
PrUp[l_] := (k - l) * (t - l) / (k * (n - k));
Do[Print["Pr(", i - 1, "|", i, ")=", PrDown[i]],
  {i, a, 1, -1}];
Print["and"];
Do[Print["Pr(", i + 1, "|", i, ")=", PrUp[i]],
  {i, a - 1, 1, -1}]
```

Pr(2|3)=$\frac{1}{4}$

Pr(1|2)=$\frac{5}{33}$

Pr(1|2)=$\frac{3}{44}$

and

Pr(3|2)=$\frac{5}{66}$

Pr(2|1)=$\frac{1}{6}$

Заметим, что вероятность удачного обмена тем меньше, чем меньше величина l.

Лемма 11.3
Пусть через N_l, $1 \le l \le a$, обозначено среднее число обменов, необходимых для того, чтобы от состояния с l ошибками перейти к состоянию с $l-1$ ошибками. Числа N_l можно вычислять рекурсивно по формулам

$$N_a = \frac{1}{\Pr(a-1|a)} = \frac{1}{1-\Pr(a|a)}, \tag{11.9}$$

$$N_{l-1} = \frac{1+\Pr(l|l-1)N_l}{\Pr(l-2|l-1)}. \tag{11.10}$$

Доказательство. Первое равенство в (11.9) непосредственно следует из определения $\Pr(a-1|a)$. Второе равенство следует из (11.8).

Для того чтобы показать (11.10), заметим, что в состоянии $l-1$ есть три возможных пути, по которым может пойти алгоритм.

i) С вероятностью $\Pr(l-2|l-1)$ он за один шаг перейдет в состояние $l-2$.

ii) С вероятностью $\Pr(l-1|l-1)$ он останется в состоянии $l-1$, тогда алгоритм достигнет состояния $l-2$ в среднем за $1+N_{l-1}$ шагов.

iii) С вероятностью $\Pr(l|l-1)$ он перейдет в состояние l, тогда алгоритм достигнет состояния $l-2$ в среднем за $1+N_{l-1}+N_l$ шагов.

Описанное доказывает следующее рекуррентное соотношение

$$N_{l-1} = \Pr(l-2|l-1)\cdot 1 + \Pr(l-1|l-1)\cdot(1+N_{l-1}) + \Pr(l|l-1)\cdot(1+N_{l-1}+N_l),$$

которое сводится к равенству (11.10), поскольку $\Pr(l-2|l-1) = 1 - \Pr(l-1|l-1) - \Pr(l|l-1)$.

■

Заметим, что при вычислении величин N_l роль играют лишь вероятности вида $\Pr(i-1|i)$.

Пример 11.3 (часть 2)

Продолжая пример 11.3, *мы видим, что значения* N_l *можно рекурсивно вычислить по формулам* (11.9) *и* (11.10).

```
Numb[a] = 1 / PrDown[a];
Do[Numb[i-1] = (1 + PrUp[i - 1] * Numb[i]) / PrDown[i - 1],
  {i, a, 2, -1}]
Do[Print["Numb(", i, ")=", Numb[i]], {i, a, 1, -1}]
```

```
Numb(3)=4
Numb(2)=43/5
Numb(1)=1606/45
```

Теорема 11.4
Среднее количество обменов в алгоритме побитового обмена при нахождении k компонент без ошибок равно

$$\sum_{j=1}^{a} \frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}} \sum_{l=1}^{j} N_l. \qquad (11.11)$$

Доказательство. Среднее число шагов, требуемых для достижения состояния 0 из состояния j, $1 \le j \le a$, можно подсчитать как среднее число шагов, требуемых для достижения состояния $j-1$ из j, плюс среднее число шагов, требуемых для достижения $j-2$ из $j-1$, и т.д. Этим объясняется внутренняя сумма в (11.11):

$$N_j + N_{j-1} + \ldots + N_1.$$

Вероятность того, что процесс начнется с состояния j, равна вероятности того, что в выбранных наугад k компонентах содержится j ошибок. Эта вероятность равна доле тех t-подмножеств n-элементного множества, которые имеют j-элементное пересечение с данным k-подмножеством (и, соответственно, $(n-j)$-элементное пересечение с его $(n-k)$-элементным дополнением). Таким образом, эта вероятность равна

$$\frac{\binom{k}{j}\binom{n-k}{t-j}}{\binom{n}{t}}.$$

Осталось перемножить два описанных выше числа и найти сумму таких произведений для всех j.

■

Пример 11.3 (часть 3)

Из теоремы 11.4 *следует, что среднее количество обменов, необходимых для кода с параметрами* $n = 23$, $k = 12$ *и* $t = 3$ *(как было определено в примере* 11.3*) при поиске* 12 *компонент без ошибок, вычисляется так:*

```
NS = Sum[(Binomial[k, j] * Binomial[n - k, t - j] /
    Binomial[n, t]) Sum[Numb[l], {l, 1, j}], {j, 1, a}];
N[NS, 5]
```

```
37.455
```

Приведенный выше алгоритм побитового обмена значительно лучше (и асимптотически тоже) методов, описанных в разд. 11.2.3. Наиболее сильные результаты в этой области можно найти в [BaKT99].

11.4 Схема Нидеррайтера

Схема Нидеррайтера [Nied86] — это видоизменение криптосистемы Мак-Элиса. В ней та же идея применяется к проверочной матрице линейного кода. Резюме этой схемы содержится в табл. 11.2 ниже.

Итак, вновь у нас есть код Гоппы $\Gamma(p_U(x), GF(2^m))$ (см. (11.1)), который определяется известным пользователю U многочленом p_U степени t_U над полем $GF(2^m)$. Пусть H_U — проверочная матрица этого кода. Ее размеры — $(n_U - k_u) \times n_u$, где k_U — размерность кода.

Код $\Gamma(p_U(x), GF(2^m))$ исправляет t_U ошибок, значит, у каждого вектора $\underline{v}$ веса $\leq t_U$ — ровно один синдром $H_U \cdot \underline{v}$. Существующий алгоритм декодирования для кодов Гоппы эффективно находит $\underline{v}$ по его синдрому.

Как и в системе Мак-Элиса, нужно спрятать структуру кода Гоппы, содержащуюся в матрице H_U. Это произойдет, если вычислить

$$H_U^* = S_U \cdot H_U \cdot P_U, \tag{11.12}$$

где S_U — обратимая $(n_U - k_u) \times (n_U - k_u)$-матрица, а P_U — матрица перестановки порядка n_U (см. (11.4)).

Матрица H_U^* объявляется публичной вместе с числом t_U.

Если Алиса хочет послать сообщение Бобу, она находит параметры H_B^* и t_B, объявленные Бобом публичными, и представляет свое сообщение в виде двоичного вектора (столбца) $\underline{m}$ веса $\leq t_U$. Она вычисляет $\underline{v} = H_B^* \cdot \underline{m}$. Это и есть шифртекст, который Алиса посылает Бобу.

Боб сначала умножает $\underline{v}$ на S_B^{-1} слева. Он получает $\underline{v}' = S_B^{-1}\underline{v} = H_B(P_B\underline{m})$ по формуле (11.12). Поскольку вектор $P_B\underline{m}$ — это перестановка компонент вектора $\underline{m}$, его вес $\leq t_B$, значит, Боб, применив алгоритм для декодирования своего кода Гоппы, может эффективно найти $\underline{m}' = P_B\underline{m}$. Для восстановления $\underline{m}$ осталось умножить $\underline{m}'$ на P_B^{-1} слева.

ПУБЛИЧНЫЕ	H_U^* И t_U ВСЕХ ПОЛЬЗОВАТЕЛЕЙ U, H_U^* ИМЕЕТ РАЗМЕР $(n_U - k_U) \times n_U$
СЕКРЕТНЫЕ	$p_U(x)$, S_U И P_U КАЖДОГО ПОЛЬЗОВАТЕЛЯ U
СВОЙСТВО	$S_U^{-1}H_U^*P_U$ — ПРОВЕРОЧНАЯ МАТРИЦА КОДА ГОППЫ, ОПРЕДЕЛЯЕМОГО МНОГОЧЛЕНОМ $p_U(x)$ СТЕПЕНИ t_U
ФОРМАТ СООБЩЕНИЯ АЛИСЫ БОБУ	$\underline{m} \in \{0,1\}^{n_B}$, ВЕС $\underline{m} \leq t_B$
ШИФРОВАНИЕ	$\underline{v} = H_B^* \cdot \underline{m}$
ДЕШИФРОВАНИЕ	ВЫЧИСЛИТЬ $\underline{v}' = S_B^{-1} \cdot \underline{c}$; ИСПОЛЬЗУЯ АЛГОРИТМ ДЕКОДИРОВАНИЯ, НАЙТИ ТАКОЙ $\underline{m}'$, ЧТО $H_B^* \cdot \underline{m}' = \underline{v}'$; ВЫЧИСЛИТЬ $P_B^{-1} \cdot \underline{m}' = \underline{m}$

Таблица 11.2. Криптосистема Нидеррайтера.

11.5 Задачи

Задача 11.1. Какова вероятность того, что случайная $k \times n$ матрица имеет ранг k? Какова вероятность того, что $k+1$ столбцов в этой матрице имеют ранг k? Вычислите эти две вероятности для $n = 16$ и $k = 5$.

Задача 11.2. Пусть C — линейный код длины $n = 23$ и размерности $k = 12$. Допустим, что сделано не более трех ошибок. Какова сложность различных атак, описанных в разд. 11.2.3?

Задача 11.3^M. Пусть C — линейный код длины 11 и размерности 6. Допустим, что сделано две ошибки. Сколько в среднем нужно сделать обменов, следуя алг. 11.2, чтобы получить 6 компонент без ошибок?

Глава 12

Системы, основанные на задаче о рюкзаке

12.1 Рюкзачная система

12.1.1 Задача о рюкзаке

В [MerH78] Меркли и Хеллман предложили криптосистему с публичными ключами, основанную на сложности решения задачи о рюкзаке. С тех пор были предложены и другие криптосистемы, основанные на этой задаче, большинство из которых оказались небезопасны. Исключением пока является схема Кора–Райвеста, описанная в [ChoR85], но в [Vaud98] показано, что ее параметры, предложенные в [ChoR85], также не гарантируют безопасности.

Определение 12.1
Пусть $a_1, a_2, ..., a_n$ — последовательность n натуральных чисел, и S — еще одно натуральное число. Вопрос о том, имеет ли уравнение

$$x_1a_1 + x_2a_2 + ... + x_na_n = S \qquad (12.1)$$

такое решение, в котором все x_i — из $\{0,1\}$, называется *задачей об укладке рюкзака* или просто *задачей о рюкзаке.*

Отметим, что в задаче о рюкзаке не требуется искать конкретного решения уравнения (12.1), вопрос лишь в том, существует ли решение. Поиск $\{0,1\}$-решения, разумеется, не менее сложен, чем вопрос о существовании решения.

При больших n задача о рюкзаке трудна для решения. На самом деле можно показать, что эта задача NP-полна (см. [GarJ79] или краткое обсуждение в разд. 11.2.2).

Для некоторых последовательностей $\{a_i\}_{i=1}^n$ нетрудно найти $\{0,1\}$-решение уравнения (12.1) или, соответственно, показать, что такого решения не существует. Например, для последовательности $a_i = 2^{i-1}$, $1 \le i \le n$, уравнение (12.1) будет иметь решение тогда и только тогда, когда $0 \le S \le 2^n - 1$. В этом случае решение легко находится.

Есть более общий класс последовательностей $\{a_i\}_{i=1}^n$, для которых уравнение (12.1) легко решается. Это класс так называемых сверхвозрастающих последовательностей.

Последовательность $\{a_i\}_{i=1}^n$ называется *сверхвозрастающей*, если для всех k, $1 \le k \le n$,

$$\sum_{i=1}^{k-1} a_i < a_k. \tag{12.2}$$

Алг. 12.1 решает задачу о рюкзаке для сверхвозрастающих последовательностей. Он находит решение $\{x_i\}_{i=1}^n$ уравнения (12.1) для каждой правой части S, для которой есть решение. Идея алгоритма очень проста. Из $\sum_{i=1}^{n-1} a_i < a_n$ следует, что в решении

$$x_n = 1 \Longleftrightarrow S \ge a_n.$$

Теперь вычтем $x_n a_n$ из S и тем же способом найдем x_{n-1}. Таким образом, получаем рекурсивно для $k = n-1, n-2, ..., 1$

$$x_k = 1 \Longleftrightarrow (S - \sum_{i=k+1}^{n} x_i a_i) \ge a_k.$$

Если в конце оказалось, что $S - \sum_{i=1}^n x_i a_i = 0$, то решение (12.1) найдено, в противном случае уравнение (12.1) решения не имеет.

Алгоритм 12.1 (*решение задачи о рюкзаке для сверхвозрастающих последовательностей*)
input $\{a_i\}_{i=1}^n$ — сверхвозрастающая последовательность натуральных чисел,
S — целое число;
initialize $k = n$;
while $k \ge 1$ **do begin**
if $S \ge a_k$ **then** $x_k = 1$ **else** $x_k = 0$;
put $S = S - x_k \cdot a_k$;
put $k = k - 1$
end
if $S = 0$ **then print** $\{x_i\}_{i=1}^n$ **else print** "решения нет".

Пример 12.1 (часть 1)

Рассмотрим сверхвозрастающую последовательность $\{a_i\}_{i=1}^6 = \{22, 89, 345, 987, 4567, 45678\}$ *и правую часть* $S = 5665$. *Чтобы увидеть, имеет ли решение уравнение* (12.1), *применим алг.* 12.1.

Поскольку $S < a_6$, *имеем* $x_6 = 0$. *Затем, видя, что* $S \ge a_5$, *получаем* $x_5 = 1$. *Вычитаем* a_5 *из* S *и получаем* 1098, *видим, что новое значение* S *удовлетворяет неравенству* $s \ge a_4$, *значит,* $x_4 = 1$ *и т.д. Окончательным решением будет последовательность* $\{1, 1, 0, 1, 1, 0\}$.

Ниже тот же процесс записан в пакете "Mathematica". При этом используются функции Length, While, If *и* Join. *Решение* $\{x_i\}_{i=1}^n$ *формируется в результате последовательного присоединения каждого вновь найденного значения* x_i *к множеству* $\{x_{i+1}, ..., x_6\}$, $i = 6, 5, ..., 1$.

```
KnapsackForSuperIncreasingSequence[a_List, S_] :=
Module[{n, x, X, T},
     n = Length[a]; X = {}; T = S;
     While[n ≥ 1,
          If[T ≥ a[[n]], x = 1, x = 0];
          T = T - x * a[[n]];
          X = Join[{x}, X]; n = n - 1];
     If[T != 0, Print["No solution"], X]]
```

```
a = {22, 89, 345, 987, 4567, 45678}; S = 5665;
X = KnapsackForSuperIncreasingSequence[a, S]
```

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

В самом деле

```
X * a
```

```
5665
```

12.1.2 Рюкзачная система

□ Формирование рюкзачной системы

Рюкзачная система, предложенная в [MerH78], основана на том, что задача о рюкзаке решается сложно, а та же задача для сверхвозрастающих последовательностей допускает легкое решение.

Каждый пользователь U создает сверхвозрастающую последовательность $\{u_i\}_{i=1}^{n_U}$ длины n_U. Затем U выбирает такие целые числа W_U и N_U, что

$$N_U > \sum_{i=1}^{n_U} u_i \tag{12.3}$$

и

$$\text{НОД}(W_U, N_U) = 1. \tag{12.4}$$

Далее U находит числа

$$u'_i = (W_U \cdot u_i) \bmod N_U, \quad 1 \le i \le n_U, \tag{12.5}$$

и объявляет последовательность $\{u'_i\}_{i=1}^{n_U}$ публичным ключом.

Пользователь U вычисляет значение $W_U^{-1} \bmod N_U$, которое окажется полезным ему при дешифровании.

Число $W_U^{-1} \bmod N_U$ можно получить с помощью расширенного алгоритма Эвклида (алг. А.8). Действительно, поскольку $\text{НОД}(W_U, N_U) = 1$, этот алгоритм дает такие X и Y, что $1 = X \cdot W_U + Y \cdot N_U$. Из этого следует, что $X \cdot W_U \equiv 1 \pmod{N_U}$, т.е. $X = W_U^{-1}$.

Каждый пользователь хранит в секрете сверхвозрастающую последовательность $\{u_i\}_{i=1}^{n_U}$, а также числа W_U, $(W_U)^{-1}$ и N_U.

Пример 12.1 (часть 2)

Продолжим работу с параметрами из примера 12.1. *Пусть Боб выбрал* $\{b_i\}_{i=1}^{6} = \{22, 89, 345, 987, 4567, 45678\}$ *в качестве своей сверхвозрастающей последовательности. После этого он выбирает число* $N_B = 56789$, *удовлетворяющее условию* $N_B > \sum_{i=1}^{6} b_i$, *и взаимно простое с ним число* $W_B = 12345$. *Затем он вычисляет* $b'_i = (W_B \cdot b_i) \bmod N_B$. *Здесь это выполнено с применением функции* Mod *пакета "Mathematica". Чтобы проверить приведенное выше условие, требуется функция* GCD.

```
b = {22, 89, 345, 987, 4567, 45678};
WB = 12345; NB = 56789;
 6
∑  b[[i]] < NB
i=1
GCD[WB, NB] == 1
bb = Mod[WB * b, NB]
```

```
|| True
```

```
|| True
```

```
|| {44434, 19714, 56639, 31669, 44927, 36929}
```

Таким образом, $\{b'_i\}_{i=1}^{6} = \{44434, 19714, 56639, 31669, 44927, 36929\}$ — *это публичный ключ.*

Решение задачи о рюкзаке представляет некоторую сложность даже для этого маленького числа n_B *(попытайтесь сделать это для числа* 101077*).*

Число $W_B^{-1} \bmod N_B$ *можно найти с помощью функций* ExtendedGCD *и* Mod *пакета "Mathematica".*

```
WB = 12345; NB = 56789;
Mod[ExtendedGCD[WB, NB], NB]
```

```
|| {1, {39750, 3704}}
```

Из этого следует, что $W_B^{-1} = 39750$. *Действительно,*

```
WBinverse =39750;
Mod[WB * WBinverse, NB]
```

```
|| 1
```

□ Шифрование

Предположим, что пользователь Алиса хочет послать сообщение Бобу. Она отыскивает публичный ключ Боба $\{b'_i\}_{i=1}^{n_B}$. Затем она представляет свое сообщение в виде двоичного вектора $(m_1, m_2, ..., m_{n_B})$ длины n_B (или нескольких векторов такой длины, если сообщение слишком длинное). Алиса посылает Бобу шифртекст

$$C = \sum_{i=1}^{n_B} m_i \cdot b'_i. \tag{12.6}$$

Пример 12.1 (часть 3)

Продолжим работу с параметрами из примера 12.1. *Публичный ключ Боба имеет вид* $\{b'_i\}_{i=1}^{6} = \{44434, 19714, 56639, 31669, 44927, 36929\}$.

Пусть $\{m_i\}_{i=1}^{6} = \{1, 1, 0, 0, 0, 1\}$ *— сообщение Алисы. Тогда она отправит Бобу шифртекст* $\sum_{i=1}^{n_B} m_i \cdot b'_i = 101077$.

```
bb = {44434, 19714,56639, 31669, 44927, 36929};
m = {1, 1, 0, 0, 0, 1};
CipherText = m * bb
```

```
101077
```

□ Дешифрование

Сначала Боб умножает полученный шифртекст C на W_B^{-1}, а затем вычисляет остаток от деления этого произведения на N_B (оба эти параметра секретны). Это означает, что

$$W_B^{-1} \cdot C \overset{(12.6)}{\equiv} W_B^{-1} \cdot \sum_{i=1}^{n_B} m_i \cdot b'_i \overset{(12.5)}{\equiv} \sum_{i=1}^{n_B} m_i \cdot b_i \pmod{N_B}.$$

Из неравенства (12.3) следует, что $\sum_{i=1}^{n_B} m_i \cdot b_i < N_B$. Поэтому приведенное выше сравнение можно переписать в виде

$$\sum_{i=1}^{n_B} m_i \cdot b_i = (W_B^{-1} \cdot C) \mod N_B. \tag{12.7}$$

Поскольку последовательность $\{b_i\}_{i=1}^{n_B}$ — сверхвозрастающая, Боб может для нахождения сообщения $\{m_i\}_{i=1}^{n_B}$ применить алг. 12.1 с правой частью $(W_B^{-1} \cdot C) \mod N_B$.

Пример 12.1 (часть 4)

Продолжим работу с параметрами из примера 12.1. *Допустим, что Боб получил сообщение* $C = 101077$. *Сначала он вычисляет* $(W_B^{-1} \cdot C) \mod N_B$ *при* $W_B^{-1} = 39750$ *и* $N_B = 56789$.

```
CipherText = 101077;
S = Mod[WBinverse * CipherText, NB]
```

```
{45789}
```

Боб получает $S = 45789$. *Для решения уравнения* (12.1) $\sum_{i=1}^{n_B} m_i \cdot b_i = S$, *он может применить определенную ранее функцию* KnapsackForSuperIncreasingSequence.

```
b = {22, 89, 345, 987, 4567, 45678}; S = 45789;
x = KnapsackForSuperIncreasingSequence[b, S]
```

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

□ Дальнейшее обсуждение

Резюме рюкзачной криптосистемы приведено в следующей таблице.

ПУБЛИЧНЫЕ	$\{u'_i\}_{i=1}^{n_U}$ ВСЕХ ПОЛЬЗОВАТЕЛЕЙ U
СЕКРЕТНЫЕ ПОЛЬЗОВАТЕЛЯ U	$\{u_i\}_{i=1}^{n_U}$, W_U^{-1} И N_U
СВОЙСТВА	$u'_i = (W_U \cdot u_i) \mod N_U$, $\{u_i\}_{i=1}^{n_U}$ — СВЕРХВОЗРАСТАЮЩАЯ, НОД$(W_U, N_U) = 1$
СООБЩЕНИЕ ДЛЯ B	$\{m_i\}_{i=1}^{n_U}$
ШИФРОВАНИЕ	$\sum_{i=1}^{n_B} m_i \cdot u'_i = C$
ДЕШИФРОВАНИЕ	ПРИМЕНИТЬ АЛГ. 12.1 К $\{u_i\}_{i=1}^{n_U}$ И $(W_B^{-1} \cdot C) \mod N_B$

Таблица 12.1. Рюкзачная криптосистема.

Несмотря на то, что рюкзачная криптосистема непригодна для подписывания, некоторое время она имела огромную популярность. Это происходило главным образом из-за простоты ее реализации. В приложениях как шифрование, так и дешифрование можно производить с данными большого объема.

Авторы [MerH78] рекомендуют пользователям брать длину $n_U = 100$, последовательность $\{u_i\}_{i=1}^{n_U}$, удовлетворяющую условию

$$(2^i - 1) \cdot 2^{100} < u_i < 2^i \cdot 2^{100}, \quad 1 \le i \le 100,$$

(она автоматически окажется сверхвозрастающей), и такой модуль N_U, что

$$2^{101} + 1 < N_U < 2^{202}.$$

Заметим, что при этом выполняется условие (12.3).

Дополнительно пользователю U рекомендуется перед публикованием ключа $\{u'_i\}_{i=1}^{n_U}$ переставить его элементы, чтобы скрыть их порядок в исходной сверхвозрастающей последовательности. Тем самым криптоаналитик лишается информации о том, какой, например, элемент u'_i в публичном ключе получен из минимального элемента секретного ключа u_1.

Разумеется, умножение сверхвозрастающей последовательности на константу W_U по модулю N_U выполняется для того, чтобы получившаяся задача о рюкзаке выглядела как случайная. Чтобы увеличить этот эффект и тем самым увеличить безопасность криптосистемы, авторы [MerH78] советуют повторить такое умножение.

Значит, каждый пользователь U дополнительно выбирает такие числа N'_U и W'_U, что $N'_U > \sum_{i=1}^{n_U} u'_i$, $1 < W'_U < N'_U$ и НОД$(W'_U, N'_U) = 1$. Затем он вычисляет $u''_i \equiv W'_U \cdot u'_i \pmod{N'_U}$, $1 \le i \le n_U$, и объявляет публичным ключом $\{u''_i\}_{i=1}^{n_U}$ вместо $\{u'_i\}_{i=1}^{n_U}$.

Польза от повторения модульного умножения иллюстрируется в следующем примере.

Пример 12.2

Пусть $n = 3$. *Рассмотрим последовательность* $\{u_i\}_{i=1}^{3} = \{5, 10, 20\}$. *Ее умножение на* 17 *по модулю* 47 *дает* $\{u'_i\}_{i=1}^{3} = \{38, 29, 11\}$. *Умножение этой последовательности на* 3 *по модулю* 89 *дает* $\{u''_i\}_{i=1}^{3} = \{25, 87, 33\}$.

Эти вычисления можно проверить с помощью функции Mod.

```
u = {5, 10, 20}
uu = Mod[17 * u,47]
uuu = Mod[3 * u,89]
```

```
|| {5, 10, 20}
|| {38, 29, 11}
|| {25, 87, 33}
```

Невозможно найти такие целые числа W *и* N, *чтобы непосредственно перевести* $\{u_i\}_{i=1}^{3}$ *в* $\{u''_i\}_{i=1}^{3}$. *Действительно, из сравнений*

$$\begin{aligned} 5 \cdot W &\equiv 25 \pmod{N}, \\ 10 \cdot W &\equiv 87 \pmod{N} \end{aligned}$$

следует, что N *делит* $87 - 2 \times 25 = 37$. *Поскольку* 37 — *простое, это означает, что* $N = 37$. *Значит,* $W = 5$. *Однако, числа* W *и* N *не удовлетворяют третьему сравнению*

$$20 \cdot W \equiv 33 \pmod{N}.$$

Это показывает, что два модульных умножения не всегда сводятся к одному.

Приведенный выше пример демонстрирует и еще кое-что. Заметим, что второе преобразование переводит не-сверхвозрастающую последовательность $\{38, 29, 11\}$ в последовательность $\{25, 87, 33\}$, которая после переупорядочения становится сверхвозрастающей.

Из этого становится понятным, что криптоаналитику Еве для преобразования публичного ключа в сверхвозрастающую последовательность вовсе не обязательно угадывать именно те исходные числа W_U и N_U (а также W'_U и N'_U в случае выполненной итерации). Ева сможет дешифровать криптограмму, если сможет получить хоть какую-нибудь сверхвозрастающую последовательность из $\{u'_i\}_{i=1}^{n_U}$ (соответственно, из $\{u''_i\}_{i=1}^{n_U}$).

Эти наблюдения демонстрируют две важные вещи:

1) Итерация не обязательно увеличивает безопасность системы.

2) Для криптоаналитика может быть легче преобразовать публичную последовательность в сверхвозрастающую, которая отлична от исходной.

Некоторые критики рюкзачной криптосистемы не доверяли ей из-за ее линейности. Их опыт и интуиция говорили, что эта система вскоре будет взломана.

Читателю следует помнить, что общая задача о рюкзаке является NP-полной. Это, в частности, означает, что не известен алгоритм, решающий ее за полиномиальное время. Однако не было доказано, что NP-полным будет и сужение задачи о рюкзаке на последовательности, полученные из сверхвозрастающих с помощью одного модульного умножения. В 1982 г. Шамир [Sham82] показал, что вариант рюкзачной системы с одним модульным умножением может быть с высокой вероятностью взломан за полиномиальное время[1]. Эта атака была позднее обобщена другими исследователями (см. [Adle83] и [Bric85]).

В разд. 12.2 будет дано краткое описание наиболее общего метода атаки, принадлежащего Лагариасу и Одлыжко [LagO83].

12.2 L^3-атака

12.2.1 Введение

В первоначальной рюкзачной криптосистеме предполагалось, что последовательность $\{u_i\}_{i=1}^{n_U}$ — сверхвозрастающая. Однако, это не является решающим фактором для криптосистем, основанных на задаче о рюкзаке. Это лишь делает дешифрование более легким, поскольку есть алг. 12.1. Единственное существенное требование состоит в том, чтобы отображение $\{m_i\}_{i=1}^{n_U} \mapsto C$ открытого текста в шифртекст в равенстве (12.6) было взаимно однозначным.

[1] Подробное изложение метода Шамира на русском языке можно найти в книге [*Сало96]. — *Прим. перев.*

Поскольку общая задача о рюкзаке является NP-полной, не известен алгоритм, решающий ее за полиномиальное время. Но вполне возможно, что есть полиномиальный алгоритм, решающий с некоторой положительной вероятностью любую задачу из большого подкласса задач о рюкзаке. Такой алгоритм сделал бы рюкзачную систему неподходящей для практических нужд.

В этом разделе часто для рюкзака $\{u_i\}_{i=1}^n$ используется векторное обозначение $\underline{u} = (u_1, u_2, ..., u_n)$.

Перед тем, как дать краткое описание *атаки Лагариаса и Одлыжко* (также называемой *L^3-атакой*) [LagO83], введем несколько новых понятий.

Определение 12.2
Плотность $d(\underline{u})$ рюкзака $\underline{u} = (u_1, u_2, ..., u_n)$ определяется как

$$d(\underline{u}) = \frac{n}{\max_{1 \le i \le n} \log_2 u_i}.$$

Пример 12.3

Например, плотность рюкзака $\{22, 89, 345, 987, 4567, 45678\}$ *равна* $6/\log_2 45678 \approx 0.39$, *что можно проверить с помощью функций* Max, Log, Length *и* N *пакета "Mathematica".*

```
u = {22, 89, 345, 987, 4567, 45678};
N[Length[u] / Log[2, Max[u]], 2]
```

```
0.39
```

Плотность $d(\underline{u})$ служит мерой информационной избыточности рюкзачной криптосистемы. В самом деле, в числителе стоит число битов сообщения, хранящихся в сумме C рюкзака (см. 12.6). Знаменатель же является хорошим приближением для среднего числа битов, необходимых для двоичного представления C. Например, при $u_i = 2^{i-1}$, $1 \le i \le n$, плотность равна $n/(n-1) \approx 1$, как и должно быть.

Далее будет показано, что вероятность успеха атаки Лагариаса и Одлыжко против рюкзачной криптосистемы тем больше, чем меньше ее плотность.

Это может показаться слишком сильным ограничением, но стоит представлять, что никому не нравится пользоваться криптосистемой, которая может быть взломана с некоторой положительной вероятностью.

12.2.2 Решетки

Определение 12.3
Пусть $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ — множество векторов из $\mathbb{Z}^n$, линейно независимых над $\mathbb{R}$. Тогда множество всех целочисленных линейных комбинаций векторов $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ называется *целочисленной решеткой*. Определяющая целочисленную решетку формула имеет вид:

$$\Lambda = \{\sum_{i=1}^{n} \alpha_i \cdot \underline{v}_i \mid \alpha_i \in \mathbb{Z}, 1 \leq i \leq n\}$$

или

$$\Lambda = \mathbb{Z} \cdot \underline{v}_1 + \mathbb{Z} \cdot \underline{v}_2 + ... + \mathbb{Z} \cdot \underline{v}_n.$$

Говорят, что векторы $\underline{v}_1, \underline{v}_2, ..., \underline{v}_n$ образуют *базис* решетки Λ. Отметим, что такой базис у решетки не единственный. Для задания решетки порядок векторов не важен, но нам в дальнейшем он понадобится. Для обозначения упорядоченного набора векторов будем использовать запись $[\underline{v}_1, \underline{v}_2, ..., \underline{v}_n]$.

Пример 12.4

Рассмотрим решетку Λ в $\mathbb{Z}^2$ с базисом $\underline{u} = (1, 2)$ и $\underline{v} = (3, 1)$. Она состоит из всех точек вида $\alpha \cdot (3, 1) + \beta \cdot (1, 2)$, где $\alpha, \beta \in \mathbb{Z}$. Ниже нарисована часть этой решетки.

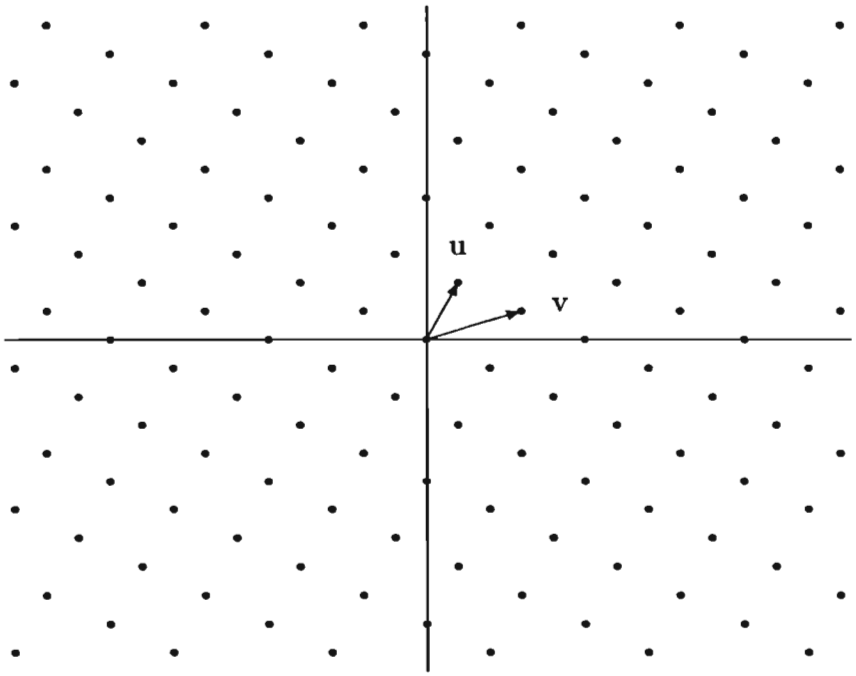


Рис. 12.1. Решетка в $\mathbb{R}^2$ с базисом $(1, 2)$ и $(3, 1)$.

Для L^3-атаки, которая будет описана позже, особенно важно найти короткий вектор решетки Λ, или еще лучше полный базис решетки Λ

из коротких векторов. Для этого нам придется изучить преобразования базиса более подробно.

Процесс ортогонализации *Грама-Шмидта* — это хорошо известный из линейной алгебры алгоритм, который преобразует базис $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ линейного пространства в ортогональный базис $\{\underline{u}_1, \underline{u}_2, ..., \underline{u}_n\}$, т.е. в базис с попарно ортогональными векторами, где $(\underline{u}_i, \underline{u}_j) = 0$ при $i \neq j$. Алгоритм действует так:

$$\begin{aligned}
\underline{u}_1 &= \underline{v}_1, \\
\underline{u}_2 &= \underline{v}_2 - \mu_{1,2}\underline{u}_1, \\
\underline{u}_3 &= \underline{v}_3 - \mu_{1,3}\underline{u}_1 - \mu_{2,3}\underline{u}_2, \\
&\vdots \\
\underline{u}_n &= \underline{v}_n - \mu_{1,n}\underline{u}_1 - \mu_{2,n}\underline{u}_2 - \ldots - \mu_{n-1,n}\underline{u}_{n-1}.
\end{aligned}$$

где

$$\mu_{i,j} = \frac{(\underline{v}_j, \underline{u}_i)}{(\underline{u}_i, \underline{u}_i)}, \quad 1 \leq i < j \leq n.$$

Пример 12.5

Для демонстрации процесса Грама-Шмидта возьмем $\underline{v}_1 = (3, 4, 2)$, $\underline{v}_2 = (2, 5, 2)$ *и* $\underline{v}_3 = (1, 2, 6)$ *в* $\mathbb{R}^3$.

```
v1 = {3, 4, 2}; v2 = {2, 5, 2}; v3 = {1, 2, 6};
u1 = v1
u2 = v2 - ((u1 * v2) / (u1 * u1) * u1
u3 = v3 - ((u1 * v3) / (u1 * u1)) * u1 -
  ((u2 * v3) / (u2 * u2)) * u2
```

```
{3, 4, 2}
```

$$\left\{-\frac{32}{29}, \frac{25}{29}, -\frac{2}{29}\right\}$$

$$\left\{-\frac{24}{19}, -\frac{24}{19}, \frac{84}{19}\right\}$$

Это можно сделать и в пакете "Mathematica". Сначала загрузим пакет LinearAlgebra`Ortogonalization`, затем запустим GramSchmidt

```
<< LinearAlgebra`Ortogonalization`
```

```
v1 = {3, 4, 2}; v2 = {2, 5, 2}; v3 = {1, 2, 6};
{u1, u2, u3} = GramSchmidt[{v1, v2, v3}]
```

$$\left\{\left\{\frac{3}{\sqrt{29}}, \frac{4}{\sqrt{29}}, \frac{2}{\sqrt{29}}\right\}, \left\{-\frac{32}{\sqrt{1653}}, \frac{25}{\sqrt{1653}}, -\frac{2}{\sqrt{1653}}\right\}, \left\{-\frac{2}{\sqrt{57}}, -\frac{2}{\sqrt{57}}, \frac{7}{\sqrt{57}}\right\}\right\}$$

Как видно из приведенного выше примера, координаты векторов $\underline{u}_i$, $1 \leq i \leq n$, могут в общем случае быть и не целыми. Это нежелательно в контексте целочисленных решеток.

В следующем подразделе мы обсудим (целочисленный) базис для решетки Λ, который не является ортонормированным, но при этом обладает двумя другими замечательными свойствами.

12.2.3 Редуцированный базис

Пусть запись $||\underline{u}||$ обозначает стандартную эвклидову норму, или длину вектора $\underline{u}$. Тогда

$$||\underline{u}|| = (\underline{u}, \underline{u})^{1/2} = \left(\sum_{i=1}^{n} (u_i)^2\right)^{1/2}.$$

Определение 12.4
Базис $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ целочисленной решетки Λ называется *y-редуцированным*, где $1/4 < y < 1$, если ортогональный базис $\{\underline{u}_1, \underline{u}_2, ..., \underline{u}_n\}$, полученный из $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ в процессе Грама-Шмидта, удовлетворяет условиям

$$\begin{aligned} &||\underline{u}_i + \mu_{i,i-1}\underline{u}_{i-1}||^2 \geq y \cdot ||\underline{u}_{i-1}||^2, && 2 \leq i \leq n, \\ &|\mu_{i,j}| \leq 1/2, && 1 \leq i < j \leq n. \end{aligned}$$

Альтернативное определение y-редуцированного базиса можно дать следующим образом. Пусть V_k — k-мерное линейное подпространство в $\mathbb{R}^n$, натянутое на $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_k\}$, или, что эквивалентно, на $\{\underline{u}_1, \underline{u}_2, ..., \underline{u}_k\}$.

Пусть $V_k^{\perp}$ — ортогональное дополнение к V_k. Обозначим через $\underline{v}_j^{(k)}$, $k+1 \leq j \leq n$, проекцию $\underline{v}_j$ на $V_k^{\perp}$. В частности, $\underline{v}_{k+1}^{(k)} = \underline{u}_{k+1}$. Тогда можно прямо показать (см. [LagO83]), что два условия в определении 12.4 эквивалентны соотношениям

$$||\underline{v}_i^{(i-2)}||^2 \geq y \cdot ||\underline{v}_i^{(i-1)}||^2 = y \cdot ||\underline{u}_{i-1}||^2, \quad 2 \leq i \leq n, \tag{12.8}$$

и, соответственно,

$$||\underline{v}_j^{(i)} + \underline{v}_j^{(i-1)}|| \leq \frac{1}{2}||\underline{v}_i^{(i-1)}||, \quad 1 \leq i < j \leq n. \tag{12.9}$$

Условие (12.8) означает, что длина проекции $\underline{v}_i$ на $V_{i-2}^{\perp}$ не должна быть слишком маленькой по сравнению с длиной $\underline{u}_{i-1}$. Неравенство в (12.9) говорит, что проекция $\underline{v}_j$ на $V_i^{\perp}$ относительно мала.

Два этих утверждения можно понимать в том смысле, что векторы y-редуцированного базиса имеют сравнимые длины и указывают в разных направлениях.

В дальнейшем везде y будет равно 3/4. L^3-алгоритм (см. [LenLL82]) – это очень эффективный способ для нахождения y-редуцированного базиса решетки Λ, который не будет здесь описан во всех деталях (тем не менее см. разд. 12.2.5). Следующие факты взяты из [LenLL82].

Теорема 12.2
Пусть $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ — базис целочисленной решетки Λ в $\mathbb{Z}^n$ и пусть $B = \max_{1 \le i \le n} ||\underline{v}_i||$. Тогда L^3-алгоритму редукции базиса решетки для создания редуцированного базиса $\{\underline{w}_1, \underline{w}_2, ..., \underline{w}_n\}$ решетки Λ требуется около $n^6(\log B)^3$ битовых операций.

Теорема 12.3
Пусть $\{\underline{w}_1, \underline{w}_2, ..., \underline{w}_n\}$ — редуцированный базис целочисленной решетки Λ. Тогда

$$||\underline{w}_1||^2 \le 2^{n-1} \cdot \min\left\{||\underline{x}||^2 \mid \underline{x} \in \Lambda \setminus \{\underline{0}\}\right\}.$$

Фактически предложение 1.12 из [LenLL82] показывает, что никакой вектор редуцированного базиса не может быть слишком длинным.

12.2.4 L^3-атака

Теперь изложим идею L^3-атаки. Мы хотим найти решение задачи о рюкзаке $\sum_{i=1}^n x_i a_i = S$ (см. (12.1)).

Идея атаки состоит в том, чтобы преобразовать параметры задачи о рюкзаке в базис для некоторой целочисленной решетки Λ. Затем найти в этом базисе короткий вектор с помощью L^3-алгоритма редукции базиса решетки. Есть надежда, что этот короткий вектор можно обратно преобразовать в решение $\{x_i\}_{i=1}^n$ задачи (12.1).

L^3-атака на $\sum_{i=1}^n a_i x_i = S$.

Шаг 1

Определим векторы

$$\begin{array}{lcccccc} \underline{v}_1 & = & (1, & 0, & ..., & 0, & -a_1), \\ \underline{v}_2 & = & (0, & 1, & ..., & 0, & -a_2), \\ & \vdots & & & & & \\ \underline{v}_n & = & (0, & 0, & ..., & 1, & -a_n), \\ \underline{v}_{n+1} & = & (0, & 0, & ..., & 0, & S). \end{array} \qquad (12.10)$$

Вместе они образуют базис $(n+1)$-мерной решетки Λ в $\mathbb{Z}^{n+1}$. Заметим, что для решения $\{x_i\}_{i=1}^n$ имеет место равенство

$$\sum_{i=1}^n x_i \underline{v}_i + \underline{v}_{n+1} = (x_1, x_2, ..., x_n, 0).$$

Таким образом, этот вектор имеет длину не более $\sqrt{n}$, т.е. он относительно короткий, в частности, если длина рюкзака $n = 100$, то $\| \sum_{i=1}^n x_i \underline{v}_i + \underline{v}_{n+1} \| \leq 10$.

Шаг 2

Найдем редуцированный базис $\{\underline{w}_1, \underline{w}_2, ..., \underline{w}_{n+1}\}$ решетки Λ с помощью L^3-алгоритма ([LenLL82]).

Шаг 3

Проверим, не будет ли у одного из этих $n+1$ "коротких" векторов $\underline{w}_i$, $1 \leq i \leq n+1$, последняя компонента $(\underline{w}_i)_{n+1}$ равна 0, а каждая из первых n компонент равна либо 0, либо α для некоторой константы α.

Если это окажется так, то $\frac{1}{\alpha}((\underline{w}_i)_1, (\underline{w}_i)_2, ..., (\underline{w}_i)_n)$ — решение уравнения (12.1). В таком случае — СТОП, иначе — переход на Шаг 4.

Шаг 4

Заменим S на $\sum_{i=1}^n a_i - S$ и повторим шаги 1, 2 и 3. Если при этом найдется решение $\{x'_i\}_{i=1}^n$ новой задачи о рюкзаке, то $\{x_i\}_{i=1}^n$, определяемое как $x_i = 1 - x'_i$, $1 \leq i \leq n$, будет решением исходной задачи.

Пример 12.6

Рассмотрим задачу о рюкзаке, в которой $\{a_i\}_{i=1}^{10} = \{541, 400, 259, 1059, 895, 590, 498, 973, 41, 649\}$ *и* $S = 4517$. *Сначала найдем векторы* $\underline{v}_i$, $1 \leq i \leq 11$, *как показано в* (12.10). *При этом используются функции* Transpose, Append, IdentityMatrix, Do, Table *и* MatrixForm *пакета "Mathematica".*

```
a = {541, 400, 259, 1059, 895, 590, 498, 973, 41, 649};
s = 4517;
aux = Transpose[Append[IdentityMatrix[10], - a]];
Do[v[i] = aux[[i]], {i, 1, 10}];
v[11] = Append[Table[0, {10}], s];
Table[v[i], {i, 1, 11}] // MatrixForm
```

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -541 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -400 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -259 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1059 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -895 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -590 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -498 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -973 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -41 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -649 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4517
\end{pmatrix};$$

Векторы $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_{11}\}$ *образуют базис решетки* Λ.

Затем с помощью функции LatticeReduce *пакета "Mathematica" найдем редуцированный базис.*

```
LatticeReduce[Table[v[i], {i, 1, 11}]]
```

```
{{ 1, -2, 1, 0, 0, 0, 0, 0, 0, 0, 0},
{-1, 0, -2, 1, 0, 0, 0, 0, 0, 0, 0},
{ 0, 1, -1, 1, -2, 1, 0, 0, 0, 0, 0},
{ 1, -1, -1, 0, -1, 0, 0, 1, 1, 0, -1},
{ 1, 1, -2, 0, 0, 1, 0, -1, -1, 0, 1},
{ 1, 1, -1, 0, 0, -2, 1, 0, 0, 0, 0},
{ 1, -1, 0, 0, 1, 0, -2, 0, -1, 0, 1},
{ 0, 1, 0, -1, 0, 1, -1, 0, -2, 1, 0},
{ 0, 0, -1, -1, -1, 1, 0, 1, 0, 1, 1},
{ 1, -1, 0, 0, 0, 1, 0, 0, -2, -1, 0},
{ 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0}}
```

Видно, что только в последнем из полученных векторов последняя компонента нулевая, а первые 10 *компонент принимают* 2 *значения. Одно из этих значений* 0, *а другое* 1. *Легко убедиться, что* $\{x_i\}_{i=1}^{10} = \{1,1,0,1,1,0,0,1,0,1\}$ *— решение для* $\sum_{i=1}^{n} a_i x_i = S$.

```
x = {1, 1, 0, 1, 1, 0, 0, 1, 0, 1};
a . x == S
```

```
True
```

Время выполнения первого и третьего шагов L^3-атаки незначительно. Следовательно, основное время работы этого алгоритма занимает (двукратный) прогон L^3-алгоритма редукции базиса решетки, время работы которого приведено в теореме 12.2. Нет гарантии, что L^3-алгоритм решает задачу о рюкзаке. Однако авторами [LagO83] дается следующий анализ L^3-алгоритма.

Теорема 12.4
Пусть $B \geq 2^{(1+\beta)n^2}$, где $\beta > 0$ — некоторая константа, а n — длина рюкзака. Пусть $K(n, B)$ обозначает количество рюкзаков $\{a_i\}_{i=1}^n$, удовлетворяющих условиям

1) $1 \leq a_i \leq B$ для всех i, $1 \leq i \leq n$,
2) L^3-атака находит $\{0,1\}$-решение $\{x_i\}_{i=1}^n$ задачи (12.1) при любом таком значении правой части S, при котором решение вообще существует.

Тогда

$$K(n, B) = B^n(1 - \varepsilon(B)),$$

где

$$0 < \varepsilon(B) < \frac{C_1}{B^{C_2 - 3(\ln n)/n}},$$

где, в свою очередь, C_1 — некоторая константа, а $C_2 = 1 - (1+\beta)^{-1} > 0$.

Теорема 12.4 утверждает, что для любого $\beta > 0$ и достаточно большого n задача о рюкзаке решается L^3-алгоритмом для почти всех рюкзаков $\{a_i\}_{i=1}^n$ с плотностью

$$d(\{a_i\}_{i=1}^n) \leq \frac{n}{\log_2 B} \leq \frac{1}{(1+\beta)n}.$$

После некоторых дополнительных усилий ([LagO83]) это неравенство можно ослабить до

$$d(\{a_i\}_{i=1}^n) < (1-\varepsilon)\frac{1}{n \cdot \log_2 4/3}$$

для любого фиксированного $\varepsilon > 0$ и достаточно большого n. Вероятно, и это неравенство — еще не предел возможного.

12.2.5 L^3-алгоритм редукции базиса

Напомним, что *L^3-алгоритм* должен найти базис $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ для целочисленной решетки, удовлетворяющий требованиям, сформулированным в определении 12.4:

$$||\underline{u}_i + \mu_{i,i-1}\underline{u}_{i-1}||^2 \geq y \cdot ||\underline{u}_{i-1}||^2, \quad 2 \leq i \leq n,$$

$$|\mu_{i,j}| \leq 1/2, \qquad 1 \leq i < j \leq n,$$

где $\mu_{i,j} = \frac{(\underline{v}_j, \underline{u}_i)}{(\underline{u}_i, \underline{u}_i)}$.

При работе L^3-алгоритм использует следующую процедуру:

Процедура reduce$[k, l]$
input l, k, $1 \leq l < k$;
compute $\mu_{l,k}$;
if $|\mu_{l,k}| > 1/2$ **then begin**
$r = \lceil 0.5 + \mu_{l,k} \rceil$;
$\underline{v}_k := \underline{v}_k - r \cdot \underline{v}_l$
end

L^3-алгоритм работает следующим образом:

L^3-алгоритм
input $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$ — базис целочисленной решетки;
initialize $k = 2$;
while $k \leq n$ **do**
begin
reduce$[k, k-1]$;
compute $||\underline{u}_k||$, $||\underline{u}_{k-1}||$, $\mu_{k-1,k}$;
if $||\underline{u}_k||^2 < (y - \mu_{k-1,k}^2) \cdot ||\underline{u}_{k-1}||^2$
then begin
exhange $\underline{v}_k$ и $\underline{v}_{k-1}$;
$k := \max\{2, k-1\}$
end
else begin
reduce$[k, l]$ для $l = k-1, ..., 2, 1$;
$k := k + 1$
end
end

Подробности можно найти в [LenLL82]. Отметим, что в алгоритме перерабатывается только базис $\{\underline{v}_1, \underline{v}_2, ..., \underline{v}_n\}$. Ни один из векторов $\underline{u}_i$ не входит в редуцированный базис. Эти векторы лишь используются при вычислениях.

12.3 Вариант Кора–Райвеста

Схема Кора–Райвеста [ChoR85] — это криптосистема, основанная на задаче о рюкзаке, в которой секретный рюкзак, для которого задача легко решается, не преобразуется в публичный, для которого задача должна быть труднорешаемой. Система использует стандартное преобразование целых чисел в двоичные последовательности фиксированной длины. Кроме того, в ней используется фиксированная константа, выбранный и зафиксированный неприводимый многочлен, выбранный и зафиксированный примитивный элемент, фиксированная перестановка и возведение в степень в конечном поле с легко решаемой задачей логарифмирования.

В [Vaud98] показано, что предложенные в [ChoR85] параметры не обеспечивают безопасности. Автор советует вернуться к первоначально-

му предложению. Здесь мы объясним лишь основную идею схемы Кора–Райвеста.

□ Формирование системы

1) Каждый пользователь U выбирает конечное поле $GF(q)$ с легко решаемой задачей логарифмирования (и для криптоаналитика тоже). Например, алгоритм Похлига–Хеллмана, описанный в разд. 8.3.1, показывает, что этого можно добиться, выбрав q так, чтобы в разложении $q-1$ на простые множители участвовали только малые простые числа. Кроме того, характеристика p поля $GF(q)$, где $q=p^k$ для некоторого k, должна удовлетворять условию $p>k$.

Для представления $GF(q)$ пользователь U применяет случайный неприводимый многочлен $f(x)$ степени k над $\mathbb{Z}_p$. Элементы $GF(q)$ можно представлять в виде многочленов степени $<k$ с коэффициентами из $\mathbb{Z}_p$ (см. теорему В.15).

Для ясности мы опускаем индексы U у всех выбранных пользователем U объектов.

2) Пользователь U выбирает в $GF(q)$ случайный примитивный элемент α. Примитивность α означает, что каждый ненулевой элемент поля $GF(q)$ можно представить в виде некоторой степени α^i, где $0\le i<q-1$. Заметим, что α, как элемент $GF(q)$ — это многочлен степени $<k$ с коэффициентами из $\mathbb{Z}_p$.

3) Для каждого $i\in\mathbb{Z}_p$ пользователь U определяет дискретный логарифм элемента $x+i$ поля по основанию примитивного элемента α. Иначе говоря, ему необходимо найти такие показатели U_i, $i\in\mathbb{Z}_p$, что

$$\alpha^{U_i}\equiv x+i \pmod{f(x)}. \tag{12.11}$$

Это реализуемо ввиду сделанного в пункте 1) предположения.

4) Наконец, пользователь U выбирает случайную перестановку π_U множества $\{0,1,...,p-1\}$ и случайный элемент D_U, $0\le D_U<q-1$. Он находит числа

$$u_i\equiv U_{\pi(i)}+D_U \pmod{q-1} \tag{12.12}$$

и объявляет эти числа $u_0,u_1,...,u_{p-1}$ публичными вместе с числом $q=p^k$. (Напомним, что $q-1$ — это порядок мультипликативной группы поля $GF(q)$, см. теорему В.20.)

Пример 12.7 (часть 1)

Боб выбирает конечное поле $GF(7^3)$, т.е. $p=7$ и $k=3$. Неприводимый многочлен $f(x)$ степени 3 над $\mathbb{Z}_7$ можно найти с помощью функции IrreduciblePolynomial пакета "Mathematica" после загрузки пакета Algebra`FiniteFields`.

```
<< Algebra`FiniteFields`
```

```
p = 7; k = 3; q = p^k;
f = IrreduciblePolynomial[x, p, k]
```

‖ $4 + x + 2x^2 + x^3$

Итак, $f(x) = x^3 + 2x^2 + x + 4$. *Оказывается, что* $\omega = x$ *— примитивный элемент в поле* $GF(7^3)$. *Это можно проверить следующим образом. Порядком любого элемента может служить только делитель числа* $q - 1 = 7^3 - 1 = 342$ *(см. теорему B.5). Из равенства* $342 = 2 \times 3^2 \times 19$ *получаем список делителей:* 1, 2, 3, 6, 9, 18, 19, 38, 57, 114, 171, 342. *То, что порядок элемента* $\omega = x$ *не совпадает с* 1, 2, 3, 6, 9, 18, 19, 38, 57, 114 *и* 171, *можно проверить следующими вычислениями. (Используется функция GF. Заметим, что* $f343$ *представляет* $GF(7^3) = \mathbb{Z}_7[x]/f(x)$.)

```
f343 = GF[7, {4, 1, 2, 1}];
om = f343[{0, 1}];
om^18
om^114
om^171
```

‖ $\{5, 5, 5\}_7$

‖ $\{2, 0, 0\}_7$

‖ $\{6, 0, 0\}_7$

Чтобы получить случайный примитивный элемент α *в поле* $GF(7^3)$, *Боб возводит* ω *в такую степень* i, *что* $\text{НОД}(i, q - 1) = 1$ *(см. лемму B.4). Используем функции Random, GCD и While.*

```
i = q - 1;
While[GCD[i, q - 1] != 1, i = Random[Integer, {1, q - 2}]];
i
```

‖ 239

Мы нашли $i = 239$. *Случайный примитивный элемент* $\alpha = \omega^i$ *равен* $3 + 4x + 5x^2$ *и вычисляется так:*

```
a = om^i
```

‖ $\{3, 4, 5\}_7$

Из сравнения $83 \times 239 \equiv 1 \pmod{q - 1}$ *следует, что* $\omega = \alpha^{83}$.

Для определения чисел B_i, *удовлетворяющих сравнениям* $\alpha^{B_i} \equiv x + i \pmod{f(x)}$, *используем*

```
B = Table[Mod[83 * FieldInd[om + i], q - 1], {i, 0, p - 1}]
```

‖ {83, 101, 175, 90, 170, 321, 213}

Заключаем, что $B_0 = 83$, $B_1 = 101$, $B_2 = 175$, $B_3 = 90$, $B_4 = 170$, $B_5 = 321$, $B_6 = 213$.

Это можно проверить так:

```
B = {83, 101, 175, 90, 170, 321, 213};
a^B
```

```
{{0, 1, 0}_7, {1, 1, 0}_7, {2, 1, 0}_7,
{3, 1, 0}_7, {4, 1, 0}_7, {5, 1, 0}_7, {6, 1, 0}_7}
```

Бобу потребуется еще кое-что сделать. Он должен выбрать случайное число D, $0 \le D < q - 1$, *и случайную перестановку* π *множества* $\{0, 1, ..., 6\}$. *Загрузим пакет DiscreteMath'Combinatorica' и применим функцию RandomPermutation.*

```
<< DiscreteMath'Combinatorica'
```

```
RD = Random[Integer, {0, q - 2}]
pi = RandomPermutation[7]
```

```
244
```

```
{6, 3, 7, 4, 5, 2, 1}
```

Итак, $D = 244$ *и* $\pi = \{6, 3, 7, 4, 5, 2, 1\}$, *т.е.* $\pi(1) = 6$, $\pi(2) = 3$, ..., $\pi(7) = 1$. *(Здесь стоит отметить, что "Mathematica" нумерует элементы массивов, начиная с* 1, *а мы начинаем нумерацию с* 0.*)*

Публичный ключ — это последовательность, заданная в (12.12): $b_i \equiv B_{\pi(i)} + D \pmod{q-1}$. *Используем функции Table и Mod.*

```
BPerm = Table[B[pi[[i]]]], {i, 1, 7}]
b = Mod[BPerm + RD, q - 1]
```

```
{321, 175, 213, 90, 170, 101, 83}
```

```
{223, 77, 115, 334, 72, 3, 327}
```

Боб объявляет публичными $\{b_i\}_{i=0}^{6} = \{223, 77, 115, 334, 72, 3, 327\}$ *и* $k = 3$.

□ Шифрование

Предположим, что пользователь Алиса хочет послать сообщение Бобу. Она отыскивает публичные параметры Боба b_0, b_1, ..., b_{p-1} и k. Затем вычисляет $q_B = p^k$. Сообщение Алисы — это число M между 1 и $\binom{p}{k}$.

Алиса представляет (описанным ниже способом) свое сообщение в виде двоичной последовательности $m_0, m_1, ..., m_{p-1}$ длины p и веса k (среди чисел m_i ровно k единиц), т.е.

$$\sum_{i=0}^{p-1} m_i = k. \tag{12.13}$$

Алиса отправляет

$$c = \sum_{i=0}^{p-1} m_i b_i \mod q_B. \tag{12.14}$$

Пример 12.7 (часть 2)

Предположим, что пользователь Алиса хочет послать сообщение Бобу. Она отыскивает публичные параметры Боба $k = 3$ *и* $\{b_i\}_{i=0}^{6} = \{223, 77, 115, 334, 72, 3, 327\}$ *(см. пример 12.7). Она знает, что* $p = 7$ *(и* $q = 7^3 = 343$*).*

Пусть $M = 19$ *— ее сообщение (это число между 1 и* $\binom{7}{3}$*).*

Как показано ниже, его можно представить в виде двоичной последовательности $\{m_i\}_{i=0}^{6} = \{0, 1, 1, 0, 1, 0, 0\}$.

Шифртекст c, который посылает Алиса, есть $\sum_{i=0}^{6} m_i b_i$. *В данном случае это число* 264.

```
m = {0, 1, 1, 0, 1, 0, 0};
ct = m . b
```

```
|| 264
```

Существует индуктивный способ представления числа M, $1 \le M \le \binom{p}{k}$, в виде двоичной строки $m_0, m_1, ..., m_{p-1}$ длины p и веса k. При этом используется хорошо известное тождество

$$\binom{p}{k} = \binom{p-1}{k} + \binom{p-1}{k-1}.$$

Если $M > \binom{p-1}{k}$, положим $m_{p-1} = 1$ и уменьшим M на $\binom{p-1}{k}$. Эта новая величина будет лежать между 1 и $\binom{p-1}{k-1}$ и будет представима в виде двоичной строки $m_0, m_1, ..., m_{p-2}$ длины $p-1$ и веса $k-1$. Если же $M \le \binom{p-1}{k}$, положим $m_{p-1} = 0$ и представим M в виде двоичной строки $m_0, m_1, ..., m_{p-2}$ длины $p-1$ и веса k.[2]

Алгоритм 12.5 (*преобразование* M *в* $m_0, m_1, ..., m_{p-1}$ *веса* k)

input M, $1 \le M \le \binom{p}{k}$;
initialize $l = k$;
for $i = 1$ **to** p **do if** $M > \binom{p-i}{l}$
then begin $m_{p-i} := 1$;
$M := M - \binom{p-i}{l}$;
$l := l - 1$
end
else $m_{p-i} := 0$

[2] Описанный процесс приводит к тому, что число M представляется в виде двоичной строки, имеющей номер M при лексикографическом (если читать строку, начиная с m_{p-1}, как это и сделано в примере) упорядочении всех строк длины p и веса k. — *Прим. перев.*

Пример 12.8

Пусть $p = 7$ *и* $k = 3$. *Тогда* $\binom{7}{3} = 35$.

Чтобы определить, в виде какой двоичной последовательности длины 7 *и веса* 3 *представляется число* $M = 19$, *воспользуемся приведенным ниже алгоритмом, в котором используются функции* Table, If, Do *и* Binomial *пакета "Mathematica".*

```
p = 7; k = 3;
Me = 19;
l = k;
m = Table[0, {i, 1, p}];
Do[If[Me > Binomial[p - i, l], {m[[i]] = 1,
Me = Me - Binomial[p - i, l], l = l - 1}], {i, 1, p}];
m
```

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

□ **Дешифрование**

Боб получает число c, которое согласно равенству (12.14) есть $\left(\sum_{i=0}^{p-1} m_i b_i\right) \bmod q_B$. Он вычисляет $C = c - k \cdot D_B$ с помощью секретного D_B (см. (12.12)).

Затем Боб вычисляет α^C. Заметим, что в поле $GF(q)$ выполняются следующие соотношения:

$$\alpha^C = \alpha^{c-k\cdot D_B} = \alpha^{\sum_{i=0}^{p-1} m_i b_i - k\cdot D_B} \overset{(12.12)}{=}$$

$$\overset{(12.12)}{=} \alpha^{\sum_{i=0}^{p-1} m_i (B_{\pi(i)} + D_B) - k\cdot D_B} \overset{(12.13)}{=} \alpha^{\sum_{i=0}^{p-1} m_i B_{\pi(i)}} =$$

$$= \prod_{i=0}^{p-1} (\alpha^{B_{\pi(i)}})^{m_i} \overset{(12.11)}{=} \prod_{i=0}^{p-1} (x + \pi(i))^{m_i} \pmod{f(x)}.$$

Это означает, что

$$\alpha^C \equiv \prod_{i=0}^{p-1} (x + \pi(i))^{m_i} \pmod{f(x)}.$$

Затем мы добавим к α^C подходящий кратный $f(x)$ многочлен, чтобы получился нормированный многочлен степени k. Итак, для некоторого $\beta \in GF(q)$ $a(x) = \alpha^C + \beta \cdot f(x)$ — нормированный многочлен степени k.

Поскольку $\prod_{i=0}^{p-1} (x+\pi(i))^{m_i} \pmod{f(x)}$ — тоже нормированный многочлен степени k, написанное выше означает, что

$$a(x) = \prod_{i=0}^{p-1} (x + \pi(i))^{m_i}.$$

Из этого следует, что $m_i = 1$, $0 \le i \le p-1$, тогда и только тогда, когда $-\pi(i)$ — корень $a(x)$.

Резюмируем процесс дешифрования в следующем алгоритме.

Алгоритм 12.6 (*дешифрование Бобом в криптосистеме Кора–Райвеста*)

input	шифртекст c;
Bob's secret	D_B, k, α, $f(x)$, π;
compute	$C = c - k \cdot D_B$ с помощью секретного. D_B (см. (12.12));
compute	α^C, где α — примитивный элемент Боба;
add	кратный $f(x)$ многочлен, чтобы получился нормированный многочлен $a(x)$ степени k;
put	$m_i = 1$, $0 \le i \le p-1$, тогда и только тогда, когда $-\pi(i)$ — корень $a(x)$.

Пример 12.7 (часть 3)

Продолжим обсуждение примера 12.7. *Предположим, что Боб получил шифртекст* $c = 264$.

Секретными параметрами Боба являются $k = 3$, $D = 244$, $\pi = \{6,3,7,4,5,2,1\}$, $f(x) = 4 + x + 2x^2 + x^3$, $\alpha = 3 + 4x + 5x^2$.

Боб вычитает $k \cdot D$ *из* c.

```
CT = Mod[ct - RD * k, q - 1]
```

```
216
```

Затем он возводит α *в степень* C. *Для того, чтобы записать это в виде многочлена, используем функцию* ElementToPolynomial *пакета "Mathematica".*

```
a^CT
u = ElementToPolynomial[a^CT, x]
```

```
{2, 1, 3}_7
```

```
2 + x + 3x^2
```

Затем Боб должен прибавить $f(x)$, *чтобы получить нормированный многочлен* $a(x)$ *степени* k. *Применим для этого функцию* PolynomialMod.

```
AX = PolynomialMod[u + f, 7]
```

```
6 + 2x + 5x^2 + x^3
```

Разложим его на множители с помощью функции Factor.

```
Factor[AX, {Modulus -> 7}]
```

```
{(2 + x) (4 + x) (6 + x)}
```

Обратную к π перестановку можно вычислить с помощью функции InversePermutation (из уже загруженного пакета DiscreteMath'Combinatorica').

```
InversePermutation[pi]
```

```
{7, 6, 2, 4, 5, 1, 3}
```

Вычтем 1, поскольку π действует не на $\{1, 2, ..., 7\}$, а на $\{0, 1, ..., 6\}$. Получим

```
InversePermutation[pi] - 1
```

```
{6, 5, 1, 3, 4, 0, 2}
```

Отсюда видно, что под действием π^{-1} числа 2, 4 и 6 переводятся в 1, 4 и 2. Иначе говоря, π отображает 1, 4 и 2 в 2, 4 и 6 соответственно.

Из этого делаем вывод, что в векторе сообщения единицы стоят в 1-й, 4-й и 2-й позициях (а нули — в 0-й, 3-й, 5-й и 6-й позициях), т.е. вектор сообщения имеет вид $\{m_i\}_{i=0}^{6} = \{0, 1, 1, 0, 1, 0, 0\}$.

Это совпадает с тем, что было зашифровано.

12.4 Задачи

Задача 12.1. Решите задачу о заполнении рюкзака, если элементы равны 333, 41, 4, 172, 19, 3, 80 и 11, а общий размер рюкзака равен 227.

Задача 12.2. Решите задачу о заполнении рюкзака, если элементы равны 31, 32, 46, 51, 63, 72, и 87, а общий размер рюкзака равен 227.

Задача 12.3^M. Публичным ключом рюкзачной криптосистемы являются числа 381, 424, 2313, 2527, 2535, 3832, 3879 и 4169. Они получены при умножении по модулю 5011 элементов сверхвозрастающей последовательности на W = 4673. Дешифруйте сообщение 11678.

Задача 12.4. Пусть $p_1, p_2, ..., p_n$ — последовательность различных простых чисел, а P — их произведение. Числа a_i, $1 \leq i \leq n$, определим как $a_i = P/p_i$. Пусть $S = \sum_{i=1}^{n} x_i \cdot a_i$, где каждый элемент x_i — это либо 1, либо 0. Найдите простой алгоритм определения по S чисел x_i, $1 \leq i \leq n$.

Задача 12.5^M. Пусть шифртекст $C = 5738$ получен при шифровании в рюкзачной криптосистеме с публичным ключом $\{u_i\}_{i=1}^{n} = \{437, 1654, 1311, 625, 1250, 1720, 663, 1420, 63, 319\}$. Примените L^3-алгоритм для поиска открытого текста.

Задача 12.6. Какое число будет представлено в виде двоичного вектора $(1,1,0,1,1,0,1,0,1,1)$ по алгоритму 12.5?

Задача 12.7^M. Проработайте полностью (включая шифрование и дешифрование) пример криптосистемы Кора–Райвеста с параметрами $p = 11$, $k = 2$.

Глава 13

Хэш-коды и техника аутентификации

13.1 Введение

В разделе 1.1 мы упомянули конфиденциальность (секретность) как первую причину, согласно которой люди используют криптосистемы. Конечно, эта цель очень важна и приводит к интересным математическим вопросам, но для огромного большинства пользователей секретность данных не является главной проблемой.

С другой стороны, аутентификация (установление подлинности) и целостность данных почти всегда существенны. Подумайте, например, о получателях файлов данных, сообщений по электронной почте или по факсу и т.п. Нарушение конфиденциальности приносит (как правило) мало вреда, но значительную опасность представляет ситуация, когда кто-нибудь способен подделать файлы данных.

При изучении схем аутентификации нужно различать следующие возможности:

i) Желательна безусловная безопасность или просто вычислительная безопасность?

ii) Доверяют различные стороны друг другу или нет?

iii) Имеется ли обоюдно доверенная третья сторона?

iv) Типичные файлы данных очень длинны или, скорее, коротки?

v) Важна ли также конфиденциальность?

vi) Предназначена ли система для многократного использования или только на один раз?

В особенности первые два различия приводят к полностью отличным друг от друга областям исследований. Главной темой разд. 13.3 являются схемы аутентификации, обладающие *безусловной безопасностью* (или абсолютной стойкостью). Это означает, что даже при неограниченных вычислительных возможностях противник не может взломать систему.

Такие схемы обычно называют *кодами аутентификации*, а один их частный подкласс называют *A-кодами*.

Вычислительно безопасные системы базируются на некоторых математических допущениях наподобие (практической) невозможности факторизовать большие числа или брать дискретные логарифмы[1]. Эти системы носят название *схем цифровых подписей* и уже обсуждались в подразд. 8.1.2, 8.2.1, 8.2.2 и 9.1.4.

В случае, когда файл — очень длинный и его конфиденциальность несущественна, имеется весьма общая техника присоединения к нему доказательства аутентичности или целостности: посылается файл с добавлением относительно короткой последовательности (например, 100–200) битов, которая сложным образом зависит от всех битов исходного сообщения. Этот хвост должен доказывать, что сообщение на самом деле пришло от предполагаемого отправителя и что его содержание не было изменено.

Стандартный путь осуществления этого — *хэшировать*[2] файл криптографически безопасным способом в короткую последовательность и вычислить подпись на этом хэш-значении. Подпись хэш-значения добавляется к исходному файлу. Если реализация схемы аутентификации работает медленно (как в случае схем цифровой подписи), такой двухшаговый подход оказывается очень практичным.

Во многих приложениях хэш-функции используют секретный ключ, разделяемый отправителем и получателем. Такие системы, называемые по-английски MAC (от Message Authentication Code), не обладают безусловной безопасностью, потому что кто-нибудь, имеющий неограниченные вычислительные ресурсы, может в принципе перебрать все ключи.

Хэш-функции и MAC'и являются предметом разд. 13.2.

13.2 Хэш-функции и MAC'и

Мы не намерены давать формальное описание различных типов хэш-кодов. Для наших целей достаточно общего понимания этих кодов и их свойств.

Функция хэширования (хэш-функция или хэш-код) — это отображение h из множества $\mathcal{A}^*$ всех последовательностей символов из алфавита $\mathcal{A}$ в $\mathcal{A}^m$, где m — некоторое фиксированное натуральное число. Таким образом, каждая последовательность (произвольной длины) над $\mathcal{A}$ отображается в последовательность длины m над $\mathcal{A}$. В типичных приложениях $\mathcal{A} = \{0, 1\}$, а m принимает значение где-то между 64 и 256.

Так как обычно желательна очень быстрая реализация хэш-функции, мы также требуем, чтобы хэш-значения для любых последовательностей над $\mathcal{A}$ вычислялись легко.

[1] См. [MeOoV97], где вычленяются пять классов безопасности криптосистем. Отметим еще, что вычислительно безопасные криптосистемы часто называют *практически безопасными*.— *Прим. ред.*

[2] От английского hash в смысле "мешанина, путаница".— *Прим. ред.*

Чтобы сделать хэш-функцию криптографически безопасной, обычно требуют выполнения одного или более из следующих условий.

H1. Хэш-функция h — односторонняя функция (см. подразд. 7.1.2), т.е. почти для всех выходов b вычислительно невозможно найти вход $a \in \mathcal{A}^*$, такой, что $b = h(a)$.

H2. Хэш-функция h *слабо сопротивляется коллизиям.* Это означает, что для заданного значения a вычислительно невозможно найти второе значение $a' \in \mathcal{A}^*$, $a' \neq a$, такое, что $h(a') = h(a)$.

H3. Хэш-функция h *сильно сопротивляется коллизиям.* Это означает, что вычислительно невозможно найти пару значений $a, a' \in \mathcal{A}^*$, $a \neq a'$, такую, что $h(a) = h(a')$.

Следствия этих требований должны быть понятны читателю. Например, H2 означает, что если хэш-значение $h(a)$ на файле a защищено цифровой подписью, то последний нельзя заменить другим файлом a' с тем же хэш-значением — просто потому, что такой a' невозможно найти. Условие H3 еще сильнее: оно дает возможность убедить судью, что система была скомпрометирована.

Пример 13.1

Рассмотрим $m = 1$ *и* $\mathcal{A} = \mathbb{Z}_n$. *При хэшировании последовательности* $a = (a_0, a_1, \ldots, a_l)$ *просто берется* $b = \left(\sum_{i=0}^{l} a_i\right) \bmod n$. *Это хэш-значение зависит от всех символов в* a *и легко вычисляется, но не выполняется ни одно из условий* $H1$–$H3$.

Пример 13.2

Снова рассмотрим $m = 1$ *и* $\mathcal{A} = \mathbb{Z}_n$. *Для хэширования последовательности* $a = (a_0, a_1, \ldots, a_l)$ *вычисляется* $b = \left(\sum_{i=0}^{l} a_i\right)^2 \bmod n$. *Если* n *— большое составное число, то условие* $H1$ *выполняется, поскольку вычисление квадратных корней по модулю такого целого* n *считается невозможным (см. теорему* 9.18*).*

С помощью функций Mod *и* Length *пакета "Mathematica" эта хэш-функция легко вычисляется.*

```
h[inputfile_list, nn_Integer] :=
  Mod[(Sum[inputfile[[i]], {i, 1, Length[inputfile]}])^2, nn]
n = 989;
in = {189, 632, 900, 722, 349};
h[in, n]
```

```
955
```

Условия H2 и H3 не выполняются, так как $-a$ имеет то же хэш-значение, что и a. Кроме того, хэш-значение сохраняется, когда одна из координат увеличивается, а другая уменьшается на одно и то же число.

```
alternative = Mod[-in, n]
h[alternative, n]
```

```
{800, 357, 89, 267, 640}
```

```
955
```

Даже когда хэш-функция удовлетворяет условиям H1–H3, остается возможность перехватить передачу $(a, h(a))$ и заменить это другим файлом $(a', h(a'))$. По этой причине иногда желательно ввести секретный ключ, разделяемый отправителем и получателем. При этом хэш-функция обретает название *код аутентификации сообщений* (MAC) и является отображением из $\mathcal{A}^* \times \mathcal{K}$ в $\mathcal{A}^m$, где $\mathcal{K}$ — пространство ключей, в точности, как в традиционных криптосистемах.

Пример 13.3

Пусть $m = 64$ и $\mathcal{A} = \mathbb{Z}_2$. Через $DES_k(a)$ обозначим результат DES-шифрования блока a длины m относительно ключа k. Допустим, что Алиса и Боб разделяют ключ k.

Рассмотрим теперь бинарный файл $\{a_1, a_2, \ldots, a_l\}$ длины l, который Алиса намерена переслать Бобу. Сначала Алиса добавляет достаточное число нулей, чтобы длина файла стала кратной 64. Пусть L — эта новая длина. Для вычисления хэш-значения на $\{a_1, a_2, \ldots, a_L\}$ Алиса выполняет следующий алгоритм.

Алгоритм 13.1 (*использование DES в качестве MAC*)

input двоичная цепочка $\{a_1, a_2, \ldots, a_L\}$, где $64|L$;

initialize $h = \{\overbrace{0, 0, \ldots, 0}^{64}\}$;

for $i = 0$ **to** $(L/64) - 1$ **do**

$h = \mathrm{DES}_k(h \oplus \{a_{64i+1}, a_{64i+2}, \ldots, a_{64i+64}\})$;

output хэш-значение h.

Получатель дублирует приведенные выше вычисления для проверки того, что файл не был изменен и что он на самом деле был послан Алисой.

Конечно, в этом примере вместо DES можно было бы использовать любой другой блочный шифр.

Можно также использовать блочный шифр в качестве бесключевой хэш-функции. С этой целью ключ делается публичным параметром.

При использовании блочного шифра для целей аутентификации неявно предполагается, что при фиксированном ключе шифр работает как случайная перестановка символов входной цепочки. Надеются также, что блочный шифр криптографически безопасен. В следующем разделе будут обсуждаться коды аутентификации, которые не базируются ни на каком математическом допущении.

Имеется много различных стандартов для хэш-функций. Читатель отсылается к [MeOoV97] и [Schne96][3].

13.3 Безусловно безопасные коды аутентификации

13.3.1 Основные понятия и границы

Никакая схема аутентификации не может дать абсолютной гарантии, что полученное сообщение пришло от подразумеваемого пользователя, скажем, Алисы. Например, всегда имеется малая вероятность того, что сгенерированная (случайно или нет) последовательность могла быть создана Алисой, хотя фактически это не так. Тогда другая сторона может счесть эту последовательность подлинным документом Алисы.

Отсюда вытекает, что необходимо определить и уметь вычислять вероятность успешного обмана. Однако при таких вычислениях имеется существенное различие между предполагаемой вычислительной безопасностью тех или иных проблем (как это было в криптосистемах с публичными ключами) и отсутствием вовсе каких-либо предположений такого сорта (безусловная безопасность или абсолютная стойкость). Последняя ситуация и будет темой данного раздела.

Мы будем предполагать, что Алиса и Боб доверяют друг другу и имеют общий секретный ключ. В действительности это предположение не является необходимым, но тогда должно быть введено понятие доверенной третьей стороны (наподобие арбитра). Начнем с простого примера.

Пример 13.4

*Алиса хочет послать Бобу единственный бит информации (*да *или* нет*) с помощью слова длины* 2. *Алисе и Бобу доступны* 4 *возможных ключа, причем партнеры используют матрицу из таблицы* 13.1.

Например, относительно третьего ключа сообщение 1 *будет послано как слово* 11. *Вероятность того, что кто-то другой может имперсонифицировать Алису, равна* 1/2, *поскольку относительно общего секретного ключа Алисы и Боба только два из четырех слов* 00, 01, 10, 11 *могут быть действительно переданы.*

Противник Ева, при попытке подменить переданное сообщение иным

[3]Российскому криптологу необходимо знать и отечественный стандарт хэш-функции *[ГОСТХ94], который используется в стандартах цифровой подписи *[ГОСТП94] и *[ГОСТ01]. — *Прим. ред.*

ключ/код	00	01	10	11
1	0	1	–	–
2	1	–	0	–
3	–	0	–	1
4	–	–	1	0

Таблица 13.1. Код аутентификации для двух сообщений.

сообщением, знает, что могут быть использованы только два ключа, но не знает, который именно. Поэтому вероятность успешной подмены также равна $1/2$. *Например, если Ева перехватывает* 01, *то она знает, что было послано либо сообщение* 1 *(относительно ключа* 1*), либо сообщение* 0 *(относительно ключа* 3*). В первом случае она должна передать* 00, *а во втором —* 11 *и, значит, ее успех имеет вероятность* $1/2$.

Эта схема обеспечивает и секретность, потому что любое переданное слово может порождаться как сообщением 0, *так и сообщением* 1 *(оба события имеют вероятность* $1/2$*).*

Общее определение кода аутентификации таково (здесь мы отклоняемся от стандартных обозначений из теории кодов аутентификации, чтобы избежать смешения со стандартными обозначениями из теории кодов, исправляющих ошибки)[4]:

Определение 13.1
Код аутентификации — это тройка $(\mathcal{M}, \mathcal{K}, \mathcal{C})$ вместе с отображением $f : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$, таким, что для всех $m, m' \in \mathcal{M}$ и всех $k \in \mathcal{K}$

$$f_k(m) = f_k(m') \Longrightarrow m = m'. \tag{13.1}$$

Множество $\mathcal{M}$ называется *множеством сообщений*, $\mathcal{K}$ — *множеством ключей*, $\mathcal{C}$ — *множеством кодовых слов.*

Код аутентификации можно описать таблицей U, в которой строки индексированы ключами k из $\mathcal{K}$, столбцы — кодовыми словами c из $\mathcal{C}$, а в клетке с индексами (k, c) стоит либо такое $m \in \mathcal{M}$, что $f_k(m) = c$ (если оно существует; тогда в силу (13.1) оно единственно), либо стоит прочерк, если такого m нет. Эту таблицу мы будем называть *матрицей аутентификации* данного кода.

В примере 13.4 $\mathcal{M} = \{0, 1\}$, $\mathcal{K} = \{1, 2, 3, 4\}$ и $\mathcal{C} = \{00, 01, 10, 11\}$. Матрица аутентификации этого кода задана в табл. 13.1.

Условие (13.1) означает, что f_k инъективно для каждого ключа k.

Когда Боб получает кодовое слово $c \in \mathcal{C}$ от Алисы, он принимает c как подписанную версию сообщения $m \in \mathcal{M}$, где m однозначно определено равенством $f_k(m) = c$. Здесь k — ключ, уже согласованный Алисой

[4] В терминах, вводимых ниже, слово "множество" часто заменяется словом "пространство"; мы послушно следуем автору.— *Прим. ред.*

и Бобом. Чтобы система была практичной, f_k должно быть легко обратимым для каждого ключа k. С этой целью структуру f_k (и $\mathcal{C}$) часто значительно упрощают.

Определение 13.2
A-код — это тройка $(\mathcal{M}, \mathcal{K}, \mathcal{T})$ вместе с отображением $g : \mathcal{M} \times \mathcal{K} \to \mathcal{T}$. При данном ключе $k \in \mathcal{K}$ сообщение m передается в виде (m, t), где $t = g_k(m)$ называется *аутентификатором* сообщения m.

Беря $f_k(m) = (m, g_k(m))$ и $\mathcal{C} = \mathcal{M} \times \mathcal{T}$, мы видим, что А-код — это частный случай кода аутентификации.

Хороший код аутентификации строится таким образом, чтобы подложные слова $\hat{c}$ равномерно распределялись над $\mathcal{C}$, тогда как подмножество слов, ожидаемых легитимным получателем, знающим общий ключ $k \in \mathcal{K}$, было лишь [небольшой] частью этого множества.

Поэтому при создании кода аутентификации цель состоит в том, чтобы не только Боб, но и арбитр мог проверить аутентичность правильно созданного c (в случае А-кода проверяется, что $g_k(m) = t$, а в случае произвольного кода аутентификации проверяется, что c лежит в образе отображения f_k), но имперсонификатор, который не знает ключа, имел бы лишь малую вероятность получить допустимое слово $\hat{c}$. Атака имперсонификатора называется *атакой имперсонификации*[5].

Вышесказанное должно оставаться верным и в случае, когда противник хочет подменить подлинное кодовое слово c (полученное с правильным ключом) другим словом $\hat{c}$, которое представляет иное сообщение. Этот сорт атаки называют *атакой подмены*. Заметим, что в этом случае противнику доступна некоторая информация о ключе. Мы не будем обсуждать системы, в которых один и тот же ключ может безопасно использоваться легитимными сторонами более одного раза.

В последующих определениях мы будем предполагать, что как ключи, так и сообщения выбираются случайно из множеств $\mathcal{K}$ и, соответственно, $\mathcal{M}$ с равномерным распределением вероятностей.

Допустим, что Алиса и Боб используют некий общий код аутентификации. Чтобы максимизировать вероятность успешной имперсонификации, противник не может сделать ничего лучшего, нежели выбрать и послать кодовое слово $c \in \mathcal{C}$, которое имеет наибольшую вероятность быть принятым легитимным получателем. В этом случае кодовое слово c должно лежать в образе отображений f_k для максимального числа ключей $k \in \mathcal{K}$.

Говоря иными словами, в матрице аутентификации ищется столбец, содержащий максимальное число не-прочерков [т.е. минимальное число прочерков]. Тогда посылается индекс c этого столбца.

[5] В российской криптографической литературе имперсонификацию часто называют "имитацией". — *Прим. ред.*

Определение 13.3
Вероятность P_I — это максимальная вероятность успешной атаки имперсонификации, т.е.

$$P_I = \max_{c \in \mathcal{C}} \frac{|\{k \in \mathcal{K} | c \in f_k(\mathcal{M})\}|}{|\mathcal{K}|}. \tag{13.2}$$

В примере 13.4 каждое кодовое слово является образом какого-то сообщения относительно ровно двух из четырех ключей (в каждом столбце из 4 клеток — 2 не-прочерка). Поэтому $P_I = 2/4 = 1/2$.

В случае атаки подмены перехватывается кодовое слово $c \in \mathcal{C}$. Это сужает множество ключей, которые могли быть использованы отправителем и получателем, до множества $\{k \in \mathcal{K} | c \in f_k(\mathcal{M})\}$. Тогда наилучшая атака для противника — искать среди кодовых слов, допустимых для данных ключей, слово, которое встречается наиболее часто.

Говоря иными словами, в матрице аутентификации данного кода отыскивается столбец с перехваченным индексом c и из матрицы удаляются все строки, содержащие прочерк в этом столбце (это строки, индексированные ключами, которые не могли быть использованы). Удаляется также столбец с индексом c. Среди оставшихся столбцов ищется столбец с наибольшим числом не-прочерков. Индекс c' этого столбца и будет подменой для c.

Определение 13.4
Вероятность P_S — это максимальная вероятность успешной атаки подмены, т.е.

$$P_S = \max_{c,c' \in \mathcal{C}, c \neq c'} \frac{|\{k \in \mathcal{K} | c, c' \in f_k(\mathcal{M})\}|}{|\{k \in \mathcal{K} | c \in f_k(\mathcal{M})\}|}. \tag{13.3}$$

В примере 13.4 каждое кодовое слово является образом какого-то сообщения относительно ровно двух из четырех ключей. Для каждого из этих двух ключей различные возможные сообщения отображаются в различные кодовые слова. Поэтому $P_S = 1/2$.

Максимальная из двух вероятностей (13.2) и (13.3) часто называется вероятностью успешного *обмана* (deception). Она дается формулой

$$P_D = \max\{P_I, P_S\}. \tag{13.4}$$

Поскольку функция аутентификации f_k инъективна для каждого $k \in \mathcal{K}$, ровно $|\mathcal{M}|$ кодовых слов должны быть аутентичны для любого данного ключа. Другими словами, каждая строка матрицы аутентификации U кода аутентификации содержит ровно $|\mathcal{M}|$ не-прочерков. Из того, что U имеет $|\mathcal{K}|$ строк и $|\mathcal{C}|$ столбцов, следует, что среднее число не-прочерков на столбец из U равно $|\mathcal{K}| \times |\mathcal{M}|/|\mathcal{C}|$. Таким образом, максимальная доля не-прочерков на столбец не меньше $|\mathcal{M}|/|\mathcal{C}|$. Это доказывает следующую теорему.

Теорема 13.2
Максимальная вероятность P_I успешной имперсонификации в схеме аутентификации для кода $(\mathcal{M}, \mathcal{K}, \mathcal{C})$ удовлетворяет неравенству $P_I \geq |\mathcal{M}|/|\mathcal{C}|$.

Аналогично предыдущему, в случае атаки подмены сужение матрицы аутентификации U на строки, которые в столбце с индексом c, где c — перехваченное кодовое слово, содержат не-прочерки, состоит из $|\{k \in \mathcal{K} | c \in f_k(\mathcal{M})\}|$ строк. После удаления столбца с индексом c новое сужение имеет $|\mathcal{C}| - 1$ столбцов и $|\mathcal{M}| - 1$ не-прочерков в каждой строке. Таким образом, среднее значение относительной частоты не-прочерков в последнем сужении равно $(|\mathcal{M}| - 1)/(|\mathcal{C}| - 1)$. Это дает следующую границу.

Теорема 13.3
Максимальная вероятность P_S успешной подмены в схеме аутентификации для $(\mathcal{M}, \mathcal{K}, \mathcal{C})$ удовлетворяет неравенству

$$P_S \geq \frac{|\mathcal{M}| - 1}{|\mathcal{C}| - 1}.$$

Даже если сообщения и ключи распределены неравномерно на пространствах сообщений и ключей соответственно, то все еще можно вывести нижние границы для P_I, P_S и P_D. В этих оценках появляются функции, которые обсуждались в гл. 5. Доказательства следующих двух теорем заинтересованный читатель может найти в [Joha94b].

Теорема 13.4
Пусть M, K и C — случайные переменные, определенные на $\mathcal{M}$, $\mathcal{K}$ и $\mathcal{C}$ соответственно, связанные отображением $f : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$, удовлетворяющим (13.1). Далее, пусть $H(X|Y)$ и $I(X;Y)$ обозначают функцию условной энтропии и, соответственно, функцию взаимной информации. Тогда

$$P_I \geq 2^{-I(C;K)}, \tag{13.5}$$

$$P_S \geq 2^{-H(K|C)}, \tag{13.6}$$

$$P_D \geq \frac{1}{\sqrt{|\mathcal{K}|}}. \tag{13.7}$$

Граница в неравенстве (13.7) называется *границей квадратного корня*. Коды аутентификации, достигающие этой границы, называют *совершенными*.

Теорема 13.5
Необходимым условием совершенности кода аутентификации служит неравенство

$$|\mathcal{M}| \leq \sqrt{|\mathcal{K}|} + 1.$$

Для дальнейшего изучения кодов аутентификации рекомендуем читателю работы [GilMS74], [MeOoV97], [Schne96] и [Simm92].

13.3.2 Конструкция проективной плоскости

В [GilMS74] можно найти изящное описание одной совершенной схемы аутентификации. Прежде, чем объяснить эту схему, мы должны рассказать, что такое проективная плоскость.

□ Конечная проективная плоскость

Проективная плоскость — это некий геометрический объект, который несколько отличается от плоскостей в обычной эвклидовой геометрии. Проективная плоскость определяется формальным путем посредством множества аксиом, которые, среди прочего, не допускают параллельных прямых! После определения мы дадим одну конструкцию проективных плоскостей, объясняющую название "проективные".

Начнем с конечного множества $\mathcal{P}$, чьи элементы называются *точками*. Далее, $\mathcal{L}$ — это совокупность подмножеств $\ell \subset \mathcal{P}$, называемых *прямыми*. Говорят, что точка P "лежит" на прямой ℓ, если $P \in \ell$. Аналогично, две прямые могут "пересекаться" в точке и т.п.; на этом пути адаптируется вся обычная терминология из геометрии. Чтобы избежать тривиальностей, будем предполагать, что каждая прямая содержит по меньшей мере две точки ($\ell \in \mathcal{L} \Longrightarrow |\ell| \geq 2$).

Определение 13.5
Пара $(\mathcal{P}, \mathcal{L})$ называется конечной *проективной плоскостью*, если выполняются следующие аксиомы:

PP-1. Существуют четыре точки, никакие три из которых не лежат на одной прямой.

PP-2. Любая пара точек лежит на единственной прямой.

PP-3. Любая пара прямых пересекается в единственной точке.

Свойство PP-1 позволяет исключить из наших рассмотрений следующий объект (изображенный на рис. 13.1): все прямые, кроме одной, имеют мощность два и проходят через одну и ту же точку, а еще одна прямая проходит через оставшиеся точки.

Теорема 13.6
Пусть $(\mathcal{P}, \mathcal{L})$ — конечная проективная плоскость. Тогда существует константа n, называемая *порядком* этой плоскости, такая, что

PP-4. Любая прямая содержит ровно $n + 1$ точек.

PP-5. Любая точка лежит ровно на $n + 1$ прямых.

PP-6. $|\mathcal{P}| = |\mathcal{L}| = n^2 + n + 1$.

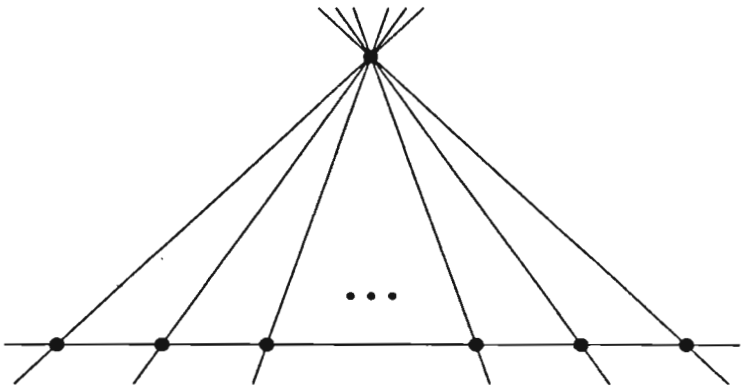


Рис. 13.1. Запрещенный объект.

Доказательство.

PP-4. На первом шаге мы должны доказать такую "лемму": каждая точка из $\mathcal{P}$ лежит по меньшей мере на трех различных прямых. Начнем с четырех точек P, Q, R и S, никакие три из которых неколлинеарны (см. PP-1). Для каждой из этих точек любая из трех других определяет единственную прямую, проходящую через них, в силу PP-2. Для точки T, не лежащей ни на какой прямой, проходящей через две из точек P, Q, R, S, "лемма" также тривиальна (каждая из этих четырех точек определяет единственную прямую, проходящую через нее и T). Мы оставляем читателю в качестве упражнения доказательство "леммы" для случая, когда точка T лежит на одной из шести прямых, проходящих через две из точек P, Q, R и S.

Рассмотрим теперь произвольную точку P. Мы знаем, что через нее проходят по крайней мере три прямые. Пусть точка Q лежит на одной из них, скажем, на прямой ℓ. Покажем, что все другие прямые, проходящие через P, имеют одну и ту же мощность. С этой целью возьмем две прямые a и b, проходящие через P и отличные от ℓ. Пусть $A_0 = P, A_1, A_2, \ldots, A_m$ — все точки на a, а $B_0 = P, B_1, B_2, \ldots, B_n$ — все точки на b (см. рис. 13.2). Мы должны показать, что $m = n$.

В силу аксиомы PP-2 для каждого $i, 0 \leq i \leq m$, существует единственная прямая, проходящая через Q и A_i. В силу аксиомы PP-3 эта прямая будет пересекать b в единственной точке; назовем ее $B_{\pi(i)}$. Единственность означает, что отображение π инъективно, откуда вытекает неравенство $m \leq n$. Меняя ролями a и b, можно заключить, что $m = n$.

Таким образом, все прямые, проходящие через P, за возможным исключением прямой, проходящей также через Q, имеют одну и ту же мощность $n+1$. Беря теперь Q на одной из других прямых, скажем, b, и повторяя приведенные выше рассуждения, заключаем, что все прямые, проходящие через P, имеют мощность $n+1$.

Пусть точка U отлична от P. Точно по той же причине, что и выше, все прямые, проходящие через U, имеют одну и ту же мощность, например, $u+1$. Однако, в силу аксиомы PP-2, одна из этих прямых проходит и через P. Это влечет, что $u = n$.

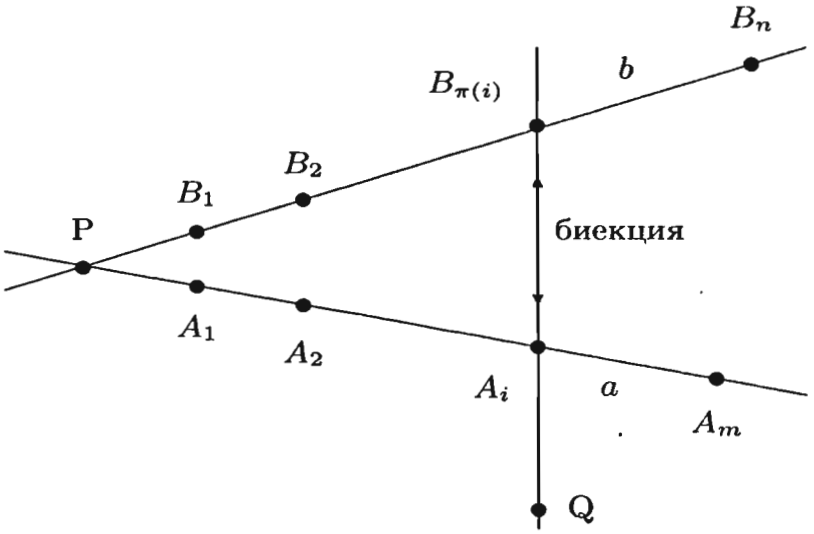


Рис. 13.2. Две прямые, проходящие через точку P.

PP-5. Рассмотрим точку P и прямую a, не содержащую P. Пусть $M_1, M_2, \ldots, M_{n+1}$ — точки на a. Каждая точка M_i вместе с P определяет, в силу аксиомы PP-2, единственную прямую, проходящую через них. Все эти прямые различны благодаря свойству единственности из PP-2. С другой стороны, любая прямая, проходящая через P, должна пересекать a в единственной точке. Заключаем, что через P проходят $n+1$ прямых.

PP-6. Рассмотрим точку P. Имеется $n+1$ прямых, содержащих P, и каждая из них содержит n других точек. В целом это дает $1+(n+1)n$ точек. В силу PP-2 других точек в $\mathcal{P}$ нет.

Аналогично, рассмотрим прямую ℓ. На ней имеется $n+1$ точек, и каждая лежит на n других прямых. В целом это дает $1+(n+1)n$ прямых. В силу PP-3 других прямых в $\mathcal{L}$ нет. (Отметим симметрию между точками и прямыми в определении 13.5.)

■

Пример 13.5

Возьмем $n=2$. Тогда $|\mathcal{P}| = |\mathcal{L}| = 7$. Каждая прямая содержит три точки, и каждая точка лежит на трех прямых. Эта проективная плоскость изображена на рис. 13.3.

На этом рисунке 7 прямых суть три внешних стороны треугольника, три медианы и овал в середине. Таким образом, $\mathcal{L}$ состоит из следующих семи прямых:

$$\begin{aligned}
\ell_1 &= \{P_1, P_2, P_3\}, & \ell_2 &= \{P_1, P_4, P_5\}\\
\ell_3 &= \{P_1, P_6, P_7\}, & \ell_4 &= \{P_2, P_4, P_6\}\\
\ell_5 &= \{P_2, P_5, P_7\}, & \ell_6 &= \{P_3, P_4, P_7\}\\
\ell_7 &= \{P_3, P_5, P_6\}.
\end{aligned}$$

Проективная плоскость порядка 2 единственна и называется плоскостью Фано.

Проективную плоскость часто описывают ее *матрицей инцидентности.* Это матрица A, строки которой индексированы прямыми $\ell \in \mathcal{L}$,

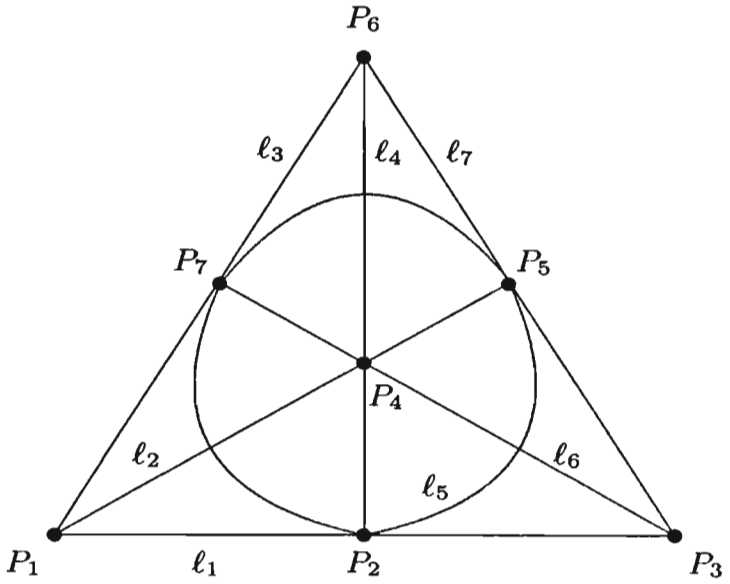


Рис. 13.3. Плоскость Фано.

столбцы — точками $P \in \mathcal{P}$ и где

$$A_{\ell,P} = \begin{cases} 1, \text{ если } P \text{ лежит на } \ell, \\ 0 \text{ в противном случае.} \end{cases}$$

Матрицей инцидентности плоскости Фано (с метками, указанными на рисунке) служит

$$\mathtt{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix};$$

Свойства из определения 13.5 и теоремы 13.6 непосредственно переводятся в следующие требования к матрице инцидентности:

PP-2. Скалярное произведение двух различных столбцов из A равно 1.

PP-3. Скалярное произведение двух различных строк из A равно 1.

PP-4. Любая строка в A содержит $n + 1$ единиц.

PP-5. Любой столбец в A содержит $n + 1$ единиц.

PP-6. A — квадратная матрица порядка $n^2 + n + 1$.

Эти свойства резюмируются формулой

$$A \cdot A^T = A^T \cdot A = n \cdot I + J, \tag{13.8}$$

где J — квадратная матрица порядка $n^2 + n + 1$, состоящая сплошь из единиц, а I — единичная матрица (того же порядка).

Для данного выше примера это можно проверить с помощью функций *Transpose* и *MatrixForm* пакета "Mathematica".

```
MatrixForm[A*Transpose[A]]
MatrixForm[Transpose[A]*A]
```

$$\begin{pmatrix} 3 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 3 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 3 \end{pmatrix} \qquad \begin{pmatrix} 3 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 3 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 3 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 3 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 3 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 3 \end{pmatrix}$$

□ Общая конструкция проективной плоскости

Имеется общая конструкция проективных плоскостей порядка q, где q — степень простого числа. Существуют и другие конструкции проективных плоскостей, но все они имеют порядок, равный степени простого числа. Было показано, что не существует проективных плоскостей порядков 6 и 10.

Через $V(3, q)$ обозначим 3-мерное векторное пространство над конечным полем $\mathrm{GF}(q)$ из q элементов. Элементами пространства служат векторы $\underline{a} = (a_1, a_2, a_3)$, где $a_i \in \mathrm{GF}(q)$. Мощность $V(3, q)$ равна q^3. Пусть $\underline{0} = (0, 0, 0)$.

Каждая прямая, проходящая через $\underline{0}$, может быть охарактеризована ненулевым вектором $\underline{a}$:

$$\{\lambda \underline{a} | \lambda \in \mathrm{GF}(q)\}. \tag{13.9}$$

Конечно, ненулевое скалярное кратное вектора $\underline{a}$ дает ту же самую прямую в $V(3, q)$. Таким образом, существуют $(q^3 - 1)/(q - 1) = q^2 + q + 1$ различных прямых, содержащих $\underline{0}$.

Аналогично, плоскость, проходящая через $\underline{0}$ в $V(3, q)$, может быть охарактеризована ненулевым вектором $\underline{u}$:

$$\{(a_1, a_2, a_3) \in V(3, q) | a_1 u_1 + a_2 u_2 + a_3 u_3 = 0\}. \tag{13.10}$$

Снова ненулевое скалярное кратное вектора $\underline{u}$ дает ту же самую плоскость в $V(3, q)$. Поэтому имеется $(q^3 - 1)/(q - 1) = q^2 + q + 1$ различных плоскостей, содержащих $\underline{0}$. Иное описание плоскости, проходящей через $\underline{0}$, — это $\{\lambda \underline{a} + \mu \underline{b} | \lambda, \mu \in \mathrm{GF}(q)\}$.

Каждая ненулевая точка на плоскости, проходящей через $\underline{0}$, определяет прямую, проходящую через $\underline{0}$. Как и выше, ненулевые скалярные кратные этой точки определяют ту же самую прямую. Следовательно, имеется $(q^2-1)/(q-1) = q+1$ прямых (содержащих $\underline{0}$) на плоскости (содержащей $\underline{0}$).

Каждую прямую $\{\lambda\underline{a}|\lambda \in \mathrm{GF}(q)\}$ можно вложить в плоскость $\{\lambda\underline{a} + \mu\underline{b}|\lambda, \mu \in \mathrm{GF}(q)\}$, выбрав любую из $q^3 - q$ точек, не лежащих на данной прямой. Конечно, не все эти плоскости различны. Отдельная плоскость, содержащая $\{\lambda\underline{a}|\lambda \in \mathrm{GF}(q)\}$, получается при выборе любой из $q^2 - q$ точек этой плоскости, не лежащих на данной прямой. Отсюда следует, что каждая прямая (содержащая $\underline{0}$) лежит в точности на $(q^3-q)/(q^2-q) = q+1$ плоскостях (содержащих $\underline{0}$).

Теорема 13.7
Пусть q — степень простого числа, $\mathcal{P}$ — множество всех прямых в $V(3,q)$, проходящих через $\underline{0}$, $\mathcal{L}$ — множество всех плоскостей в $V(3,q)$, проходящих через $\underline{0}$. Тогда $(\mathcal{P}, \mathcal{L})$ — проективная плоскость порядка q.

Замечание 1. Здесь легко запутаться: проективные точки соответствуют прямым из $V(3,q)$ (содержащим $\underline{0}$), а проективные прямые — плоскостям из $V(3,q)$ (содержащим $\underline{0}$).

Замечание 2. Отметим, что свойства PP-4, PP-5 и PP-6, указанные в теореме 13.6, уже проверены.

Доказательство теоремы 13.7. Проверим аксиомы.

PP-1. Четыре прямые, проходящие через $\underline{0}$ и одну из точек (1,0,0), (0,1,0), (0,0,1), (1,1,1), определяют четыре проективные точки в $\mathcal{P}$, никакие три из которых не лежат на одной проективной прямой. Основанием служит то, что никакие три из указанных четырех точек в $V(3,q)$ не лежат в одной плоскости, проходящей через $\underline{0}$.

PP-2. Пусть P и Q — две различные проективные точки, определяемые прямыми $\{\lambda\underline{a}|\lambda \in \mathrm{GF}(q)\}$ и $\{\lambda\underline{b}|\lambda \in \mathrm{GF}(q)\}$ в $V(3,q)$. Существует в точности одна плоскость, содержащая эти прямые, а именно $\{\lambda\underline{a} + \mu\underline{b}|\lambda, \mu \in \mathrm{GF}(q)\}$. Эта плоскость определяет единственную проективную прямую, проходящую через P и Q.

PP-3. Пусть a и b — различные проективные прямые. Они соответствуют двум плоскостям в $V(3,q)$, содержащим $\underline{0}$. Пересечением этих плоскостей является прямая, содержащая $\underline{0}$, которая определяет единственную проективную точку, общую для a и b.

■

Существует различная техника для порождения q^2+q+1 ненулевых точек в $V(3,q)$, которая приводит к различным прямым и плоскостям в $V(3,q)$, проходящим через $\underline{0}$ (см. (13.9) и (13.10)), т.е. к q^2+q+1 различным проективным точкам и, соответственно, проективным прямым.

Элегантный способ — как мы увидим в ближайшем примере — взять какой-нибудь примитивный элемент ω в $\mathrm{GF}(q^3)$, представить его

как вектор в $V(3,q)$ и выбрать в качестве нужных точек элементы $1,\omega,\ldots,\omega^{q^2+q}$. В самом деле, положим $\alpha = \omega^{q^2+q+1} = \omega^{(q^3-1)/(q-1)}$. Из того факта, что ω имеет порядок q^3-1, следует, что α имеет порядок $q-1$. Это также влечет, что $\{0,1,\alpha,\ldots,\alpha^{q-2}\} = \mathrm{GF}(q)$ (см. теорему B.29 и замечание в конце подразд. B.4.6). Поэтому для каждого $j, 1 \le j \le q-2$, точки ω^i и $\omega^{i+j(q^3-1)/(q-1)}$ в $V(3,q)$ приводят к одной и той же проективной точке u; значит, нужно рассматривать лишь $1,\omega,\ldots,\omega^{q^2+q}$.

Пример 13.6

Возьмем $q = 3$. Чтобы найти примитивный многочлен степени 3 над GF(3), нужно сначала загрузить пакет Algebra'FiniteFields' из "Mathematica", а затем применить функцию FieldIrreducible.

```
<< Algebra'FiniteFields'
```

```
m = 3; p = 3;
FieldIrreducible[GF[p, m], x]
```

```
|| 1 + 2x^2 + x^3
```

Таким образом, $\mathrm{GF}(3^3)$ описывается множеством тернарных многочленов по модулю $f(x) = x^3 + 2x^2 + 1$. Пусть $\omega \in \mathrm{GF}(3^3)$ — нуль многочлена $f(x)$. Так как $f(x)$ примитивен, ω имеет порядок 26. Это легко проверить:

```
f27 = GF[3, {1, 0, 2, 1}];
om = f27[{0, 1, 0}];
om^2
om^13
```

```
|| {0, 0, 1}_3
```

```
|| {2, 0, 0}_3
```

В данном случае элемент $\alpha = \omega^{(q^3-1)/(q-1)}$ равен $\omega^{13} = 2$. Очевидно, $\{0,1,\alpha\} = \mathrm{GF}(3)$. Таким образом, $3^2+3+1 = 13$ проективных точек можно найти, вычисляя $\omega^i, 0 \le i < 13$. В этом примере, чтобы сделать выход единообразным по форме, мы возьмем эквивалентное множество индексов $i : 1 \le i \le 13$.

```
Do[Print[om^i], {i, 1, 13}]
```

```
|| {0, 1, 0}_3
   {0, 0, 1}_3
   {2, 0, 1}_3
   {2, 2, 1}_3
   {2, 1, 1}_3
   {2, 2, 2}_3
   {1, 2, 1}_3
   {2, 1, 0}_3
```

```
{2, 2, 0}3          {0, 2, 1}3
{0, 2, 2}3          {2, 0, 0}3
{1, 0, 1}3
```

Чтобы проверить, лежит ли проективная точка $\omega^i = (a_1, a_2, a_3)$ *на проективной прямой, определяемой вектором* $\omega^j = (u_1, u_2, u_3)$ *(см.* (13.10)*), нужно выяснить, будет ли* $a_1u_1 + a_2u_2 + a_3u_3 = 0$. *В пакете "Mathematica" это можно сделать следующим образом (*[[1]] *удаляет нижний индекс в представленном выходе):*

```
i = 5; j = 12;
a = om^i[[1]]
b = om^j[[1]]
Mod[a * b, 3] == 0
```

```
{2, 2, 0}
```

```
{0, 2, 1}
```

```
False
```

Итак, мы готовы генерировать проективную плоскость порядка 3. *Представим ее посредством матрицы инцидентности.*

```
A = Table[If[Mod[(om^i[[1]])*(om^j[[1]]), 3] == 0, 1, 0],
    {i, 1, 13}, {j, 1, 13}];
MatrixForm[A]
```

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

Свойства PP-2, PP-3, PP-4 и PP-5 можно проверить вычислением (см. (13.8))

```
MatrixForm[A * Transpose[A]]
```

$$\begin{pmatrix}
4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4
\end{pmatrix}$$

□ Код аутентификации, определяемый проективной плоскостью

Определение 13.6
Пусть $(\mathcal{P}, \mathcal{L})$ — проективная плоскость, ℓ — одна из проективных прямых. Соответствующий код аутентификации определяется условиями:

$$\mathcal{M} = \ell, \qquad \mathcal{K} = \mathcal{P} \setminus \ell, \qquad \mathcal{C} = \mathcal{L} \setminus \{\ell\};$$

$f_P(Q)$ — единственная прямая, проходящая через P и Q,
где $P \in \mathcal{K}, \quad Q \in \mathcal{M}$.

Итак, множество сообщений $\mathcal{M}$ состоит из точек на ℓ, ключевое пространство состоит из всех точек, не лежащих на ℓ, множество $\mathcal{C}$ кодовых слов состоит из всех прямых, исключая ℓ.

Восстановление сообщения из полученного кодового слова производится легко: нужно просто пересечь $c = f_P(Q)$ с ℓ. Точка пересечения u будет сообщением.

То, что описанная схема определяет код аутентификации, очевидно. Параметры кода дает следующая теорема.

Теорема 13.8
Код аутентификации, определяемый проективной плоскостью порядка n, имеет параметры

$$|\mathcal{M}| = n + 1, \qquad |\mathcal{K}| = n^2, \qquad |\mathcal{C}| = n^2 + n.$$

Вероятности успеха атак имперсонификации и подмены даются равенствами

$$P_I = P_S = \frac{1}{n}.$$

Читатель может проверить эту теорему на примере плоскости Фано. Четыре точки вне ℓ (см. рис. 13.4) образуют ключевое пространство $\mathcal{K}$, три точки на ℓ — пространство сообщений $\mathcal{M}$, отличные от ℓ шесть прямых — множество $\mathcal{C}$ кодовых слов.

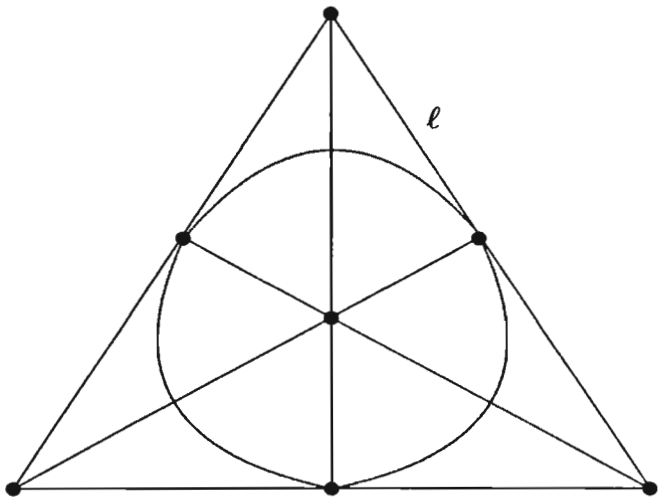


Рис. 13.4. Проверка теоремы 13.8 на примере плоскости Фано.

Доказательство теоремы 13.8. Утверждение о параметрах прямо следует из теоремы 13.6.

Для вычисления P_I заметим, что противник не может сделать ничего лучшего, как выбрать в качестве кодового слова прямую c ($c \neq \ell$), содержащую как можно больше точек вне ℓ (они являются возможными ключами). Однако число таких точек не зависит от выбора c. В силу свойства PP-4 оно равно n. Поэтому, согласно равенству (13.2),

$$P_I = \frac{n}{|\mathcal{K}|} = \frac{n}{n^2} = \frac{1}{n}.$$

Аналогично, если противник наблюдает кодовое слово c (отличное от ℓ), все еще остаются n возможных ключей (точек на c, не лежащих на ℓ). Пусть P — пересечение c и ℓ. Чтобы заменить P другим сообщением (точкой Q на ℓ), противник не может сделать ничего лучшего, как выбрать прямую d, проходящую через Q, с наибольшим множеством точек на c. Но в силу аксиомы PP-2 это число равно 1 независимо от выбора c и d, а именно — это единственная точка пересечения c и d. Поэтому, согласно равенству (13.3),

$$P_S = \frac{1}{n}.$$

■

Коды аутентификации, определяемые проективными плоскостями, совершенны, потому как P_I, P_S и P_D все равны $1/n$, а это равно $\frac{1}{\sqrt{|\mathcal{K}|}}$.

Сверх того, $|\mathcal{M}| = n+1 = \sqrt{|\mathcal{K}|}+1$ и теорема 13.5 говорит, что множество сообщений имеет максимальный размер при данном пространстве ключей.

В [Joha94a] можно найти построение кодов аутентификации с помощью последовательностей регистров сдвига. Их реализация проще, чем приведенная выше конструкция на базе проективной плоскости. Для больших множеств сообщений, например, файлов данных, более практичны коды, обсуждаемые в подразд. 13.3.4.

13.3.3 A-коды, определяемые ортогональными массивами

Определение 13.7
Ортогональный массив $\mathrm{OA}(n, k, \lambda)$ — это $k \times (\lambda \cdot n^2)$-матрица над n символами, такая, что в любых двух строках каждая возможная пара символов встречается ровно λ раз.
Число λ называется *индексом* ортогонального массива, а k — его *глубиной*.

Из этого определения следует, что каждый символ встречается ровно $\lambda \cdot n$ раз в каждой строке.

Пример 13.7 (часть 1)

Приведем пример ортогонального массива OA(4,5,1):

$$\mathtt{U} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 1 & 0 & 3 & 2 & 2 & 3 & 0 & 1 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 2 & 3 & 0 & 1 & 3 & 2 & 1 & 0 & 1 & 0 & 3 & 2 \\ 0 & 1 & 2 & 3 & 3 & 2 & 1 & 0 & 1 & 0 & 3 & 2 & 2 & 3 & 0 & 1 \end{pmatrix};$$

Следующая теорема показывает, как естественным образом ортогональные массивы определяют A-коды.

Теорема 13.9
Пусть U — ортогональный массив $\mathrm{OA}(n, k, \lambda)$. Пусть строки в U индексированы множеством $\mathcal{M}$, а столбцы — множеством $\mathcal{K}$. Далее, положим $\mathcal{T} = \{1, 2, \ldots, n\}$ и определим отображение $g : \mathcal{M} \times \mathcal{K} \to \mathcal{T}$ равенствами $g_k(m) = U_{m,k}$. Тогда g определяет A-код с параметрами $|\mathcal{M}| = k$, $|\mathcal{K}| = \lambda \cdot n^2$, $|\mathcal{T}| = n$. Сверх этого,

$$P_I = P_S = 1/n.$$

Доказательство. Параметры A-кода определяются параметрами матрицы U.

Шансы на успешную атаку имперсонификации составляют $1/n$, потому что в любой строке U каждый символ встречается одинаково часто.

Вероятность успешной атаки подмены также равна $1/n$. Причина в том, что каждый перехваченный аутентификатор встречается λ раз с каждым символом, все равно, какое сообщение перехвачено и каким сообщением его желательно заменить.

■

Пример 13.7 (часть 2)

В матрице U, определенной выше, сообщение 4 *относительно ключа* 13 *будет аутентифицировано так:*

```
m = 4; k = 13;
U[[m, k]]
```

```
1
```

Когда перехвачено сообщение 4 *с аутентификатором* 1, *известно, что ключ лежит в множестве* $\{2, 8, 11, 13\}$. *"Mathematica" может найти эти позиции с помощью функций* Flatten *и* Position*:*

```
l = Flatten[Position[U[[4]], 1]]
```

```
{2, 8, 11, 13}
```

Каждая из остальных строк на этих четырех местах содержит все четыре символа. Это можно проверить с помощью функций MatrixForm *и* Transpose. *Ниже* $[[l]]$ *дает сужение матрицы на строки, индексированные элементами списка* l.

```
SubU = Transpose[U][[l]];
MatrixForm[Transpose[SubU]]
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 3 & 2 & 0 \\ 1 & 2 & 0 & 3 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 3 & 2 \end{pmatrix}$$

Имеется очень много литературы по ортогональным массивам. См. [Hall67] или [BeJL86] относительно их конструкции, границ и результатов существования. Например, известно, что $\mathrm{OA}(q, q+1, 1)$ существуют для всех степеней q простых чисел, поскольку ортогональные массивы с этими параметрами существуют тогда и только тогда, когда существуют проективные плоскости порядка q (см. теорему 13.7 о построении проективной плоскости порядка q).

Ниже мы даем набросок доказательства этого результата.

Пусть $(\mathcal{P}, \mathcal{L})$ — проективная плоскость порядка q. Выберем в $\mathcal{L}$ произвольную прямую ℓ. Занумеруем ее точки: $P_1, P_2, \ldots, P_{q+1}$; пусть остальные точки плоскости — это $Q_1, Q_2, \ldots, Q_{q^2}$.

Далее, пусть $\mathcal{L}_i, 1 \le i \le q+1$, — совокупность всех прямых, проходящих через P_i за исключением самой ℓ. В силу свойства PP-5 каждое $\mathcal{L}_i$ имеет мощность q. Занумеруем прямые в каждом $\mathcal{L}_i$ от 1 до q.

Определим $U_{i,j}, 1 \le i \le q+1, 1 \le j \le q^2$, положив $U_{i,j} = k$, где $k, 1 \le k \le q$, — индекс единственной прямой в $\mathcal{L}_i$, которая содержит Q_j (это единственная прямая в $\mathcal{L}$, проходящая через P_i и Q_j). Тогда U является $\mathrm{OA}(q, q+1, 1)$.

Пример 13.8

Рассмотрим матрицу инцидентности A проективной плоскости порядка 3 *из примера* 13.6*:*

$$\mathtt{A} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix};$$

Определим функцию RowSwap, чтобы выполнять перестановки строк в матрице.

```
RowSwap[B_, i_, j_] := Module[{U, V},
  U = B; V = U[[i]]; U[[i]] = U[[j]]; U[[j]] = V; U];
```

Теперь мы выполним некоторые перестановки столбцов в A, чтобы получить прямую в верхней строке, все точки которой лежат слева. Используем функцию Transpose.

```
B = Transpose[A];
B = RowSwap[B, 1, 7]; B = RowSwap[B, 4, 13];
B = Transpose[B];
MatrixForm[B]
```

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

Теперь выполним ряд перестановок строк, чтобы получить хорошо выровненные подмножества $\mathcal{L}_i$ *(*$\mathcal{L}_1$ *будет располагаться в строках* $2, 3, 4$, $\mathcal{L}_2$ *— в строках* $5, 6, 7$ *и т.д.).*

```
BB = B;
BB = RowSwap[BB, 2, 8]; BB = RowSwap[BB, 6, 11];
BB = RowSwap[BB, 7, 13]; BB = RowSwap[BB, 8, 13];
MatrixForm[BB]
```

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

Последние 9 *столбцов определяют ортогональный массив* $\mathrm{OA}(3, 4, 1)$. *Например, столбец* 5 *без его первого входа имеет вид* $(1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1)$. *Этот вектор — конкатенация четырех троек, каждая из которых содержит одну* 1. *Он отображается на четыре входа из множества* $\{1, 2, 3\}$ *в зависимости от того, будет ли* 1 *первой координатой, второй или третьей; поэтому столбец* 5 *отображается в столбец* $(1, 2, 3, 3)$.

Таким образом, последние 9 *столбцов с помощью функций* Table, If *и* Do *пакета "Mathematica" отображаются на* 4×9*-матрицу:*

```
U = Table[0, {i, 1, 4}, {j, 1, 9}];
Do[b = {BB[[2 + (i - 1) * 3, j]],
    BB[[3 + (i - 1) * 3, j]], BB[[4 + (i - 1) * 3, j]]};
       U[[i, j - 4]] = If[b == {1, 0, 0},
    1, If[b == {0, 1, 0}, 2, 3]],
       {i, 1, 4}, 2, 3]],
MatrixForm[U]
```

$$\begin{pmatrix} 1 & 3 & 2 & 1 & 2 & 2 & 3 & 1 & 3 \\ 2 & 3 & 3 & 1 & 2 & 1 & 1 & 3 & 2 \\ 3 & 3 & 1 & 1 & 2 & 3 & 2 & 2 & 1 \\ 3 & 2 & 3 & 2 & 2 & 1 & 3 & 1 & 1 \end{pmatrix}$$

Это и в самом деле $\mathrm{OA}(3,4,1)$ *и, следовательно, определяет A-код с параметрами* $|\mathcal{M}| = 4$, $|\mathcal{K}| = 9$, $|\mathcal{T}| = 3$ *и* $P_I = P_S = 1/3$.

Заметим, что последние 9 *столбцов в* U *(или в* A*) можно еще переставить для получения матрицы*

$$\begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 2 & 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 3 & 2 & 3 & 2 & 1 & 2 & 1 & 3 \\ 2 & 3 & 1 & 1 & 2 & 3 & 3 & 1 & 2 \end{pmatrix}$$

13.3.4 A-коды, определяемые кодами, исправляющими ошибки

В [JohKS93] показано, как можно строить коды аутентификации, исходя из кодов, исправляющих ошибки (или EC-кодов — от Error-Correcting Codes), и обратно. В этом подразделе мы покажем, как преобразовать EC-код в A-код. Наше описание слегка отличается от оригинального.

Пусть C — произвольный $(n, |C|, d_H)$-EC-код над $\mathrm{GF}(q)$, т.е. C — подмножество n-мерного векторного пространства $V(n,q)$ над $\mathrm{GF}(q)$ с минимальным расстоянием Хэмминга d_H. Последнее означает, что все элементы в C, называемые кодовыми словами, отличаются друг от друга по меньшей мере в d_H координатах. Размерность n пространства $V(n,q)$ называют также длиной кода C.

Пусть C обладает дополнительным свойством:

$$\underline{c} \in C \Longrightarrow \underline{c} + \lambda\underline{1} \in C \qquad \text{для всех } \lambda \in \mathrm{GF}(q), \tag{13.11}$$

где $\underline{1}$ — вектор, состоящий только из единиц. Например, любой линейный код, содержащий вектор $\underline{1}$, удовлетворяет уловию (13.11). Заметим, что из (13.11) следует, что q делит мощность C.

Отношение $\sim$, определяемое на C условием

$$\underline{c} \sim \underline{c}' \Longleftrightarrow \underline{c} - \underline{c}' = \lambda\underline{1} \quad \text{для некоторого } \lambda \in \mathrm{GF}(q), \tag{13.12}$$

является отношением эквивалентности. Пусть M — подкод в C, содержащий по одному представителю из каждого класса эквивалентности. Таким образом, M имеет мощность $|C|/q$, а $C = \{\underline{m} + \lambda\underline{1} | \underline{m} \in M, \lambda \in \mathrm{GF}(q)\}$.

Пусть $\underline{m}_i, 0 \leq i < |C|/q$, — произвольное перечисление кодовых слов из M. В качестве множества $\mathcal{M}$ сообщений для кода аутентификации, который здесь строится, возьмем $\mathcal{M} = \{0, 1, \ldots, (|C|/q) - 1\}$. Это означает, что имеется взаимно однозначное соответствие между подкодом M и индексным множеством $\mathcal{M}$. Часто удобно не различать эти два множества. Поэтому, начиная с этого места, мы будем говорить о сообщении $\underline{m}_i$ вместо сообщения i.

Пример 13.9 (часть 1)

Рассмотрим бинарный линейный код C с порождающей матрицей

$$\mathtt{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix};$$

Это означает, что C состоит из 16 векторов (бинарной) линейной оболочки строк. Легко проверить, что различные кодовые слова из C различаются по меньшей мере тремя координатами. Это делает C $(7, 16, 3)$-кодом в $V(7, 2)$. Некоторые читатели могут узнать в C код Хэмминга.

Легко проверить, что вектор $\underline{1}$ лежит в C:

```
inf = {1, 1, 1, 1};
Mod[inf * C, 2]
```

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

Отсюда вытекает, что C удовлетворяет уловию (13.11).

В качестве подкода M в C возьмем все кодовые слова из C с нулевой первой координатой. Таким образом, M является линейной оболочкой трех нижних строк из G. Множество сообщений $\mathcal{M}$ отождествляется с M.

Множество ключей $\mathcal{K}$ кода аутентификации, который строится здесь, будет состоять из пар (i, λ), где $1 \leq i \leq n$ и $\lambda \in \mathrm{GF}(q)$. Таким образом, $\mathcal{K} = \{1, 2, \ldots, n\} \times \mathrm{GF}(q)$ и $|\mathcal{K}| = n \cdot q$.

Аутентификатор $g_k(\underline{m})$ сообщения $\underline{m} \in M$ относительно ключа $k = (i, \lambda)$ дается равенством

$$g_k(\underline{m}) = m_i + \lambda. \tag{13.13}$$

Поэтому множество $\mathcal{T}$ аутентификаторов просто совпадает с $\mathrm{GF}(q)$.

Теорема 13.10
Пусть C — $(n, |C|, d_H)$-код, удовлетворяющий (13.11), а M — подкод в C, содержащий по одному элементу из каждого класса эквивалентности относительно отношения (13.12). Пусть $\mathcal{K} = \{1, 2, \ldots, n\} \times \mathrm{GF}(q)$, $\mathcal{T} = \mathrm{GF}(q)$ и отображение g определяется равенством (13.13).
Тогда $(\mathcal{M}, \mathcal{K}, \mathcal{T})$ является A-кодом с параметрами

$$|\mathcal{M}| = |\mathcal{C}|/q, \qquad |\mathcal{K}| = n \cdot q, \qquad |\mathcal{T}| = q; \tag{13.14}$$

$$P_I = 1/q, \qquad P_S \leq 1 - d_H/n. \tag{13.15}$$

Замечание. Чтобы сделать P_S приемлемо низкой, нужны EC-коды с d_H, близким к n. Как мы увидим в примере 13.10, для q-арных кодов это — не проблема. Разумеется, q тоже должно быть большим.

Доказательство теоремы 13.10. Параметры в (13.14) непосредственно определяются конструкцией.

Чтобы вычислить P_I, заметим, что противник, желающий имперсонифицировать отправителя, должен найти правильный аутентификатор для своего сообщения $\underline{m}'$. Однако для каждой i-й координаты, $1 \leq i \leq n$, множество $\{m_i + \lambda | \lambda \in \mathrm{GF}(q)\}$ совпадает с $\mathrm{GF}(q)$. Иными словами, каждый символ встречается одинаково часто как аутентификатор для $\underline{m}'$. Поэтому вероятность того, что противник выберет правильный аутентификатор, равна $1/q$ независимо от выбора аутентификатора и сообщения $\underline{m}'$, которые противник попытается передать. Это доказывает, что $P_I = 1/q$.

Противник, который хочет подменить аутентифицированное сообщение $(\underline{m}, t)$ (где $t = g_k(\underline{m})$) другим аутентифицированным сообщением, знает, что использованный ключ взят из множества n возможных ключей (i, λ). Более точно, для каждой i-й координаты, $1 \leq i \leq n$, существует единственное значение λ, такое, что $m_i + \lambda = t$.

Оптимальная стратегия для противника, который хочет подменить сообщение $(\underline{m}, t)$ другим аутентифицированным сообщением, — найти такое сообщение $\underline{m}'$, $\underline{m}' \neq \underline{m}$, что $g_k(\underline{m}') = t'$ для возможно большего числа из n ключей. Символ t' — аутентификатор для $\underline{m}'$, который будет принят с наибольшим правдоподобием.

Остается показать, что t' будет принят самое большее в $n - d_H$ случаях, откуда следует, что вероятность успешной подмены равна самое большее $(n - d_H)/n = 1 - d_H/n$. Это утверждение вытекает из следующей цепочки соотношений:

$$|\{(i, \lambda) \in \{1, 2, \ldots, n\} \times \mathrm{GF}(q) | (\underline{m})_i + \lambda = t \,\&\, (\underline{m}')_i + \lambda = t'\}| =$$

$$= |\{i | 1 \leq i \leq n, (\underline{m} - \underline{m}')_i = t - t'\}| =$$

$$= n - d_H(\underline{m} - \underline{m}', (t - t')\underline{1}) \leq n - d_H,$$

поскольку $\underline{m} - \underline{m}'$ и $(t - t')\underline{1}$ — различные слова кода C ($\underline{m}$ и $\underline{m}'$ лежат в разных классах эквивалентности).

■

Пример 13.9 (часть 2)

Чтобы проиллюстрировать вторую часть доказательства, приведенного выше, мы продолжим рассмотрение кода из примера 13.9. *Если Алиса хочет послать сообщение* 7, *то она находит* $\underline{m}$ *с помощью функции* IntegerDigits *пакета "Mathematica":*

```
mes = 7;
inf = IntegerDigits[mes, 2, 4]
m = Mod[inf * G, 2]
```

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

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

(Напомним, что все сообщения имеют нулевую первую координату.)

Допустим, что Алиса и Боб имеют согласованный ключ $(3, 1)$. *Тогда Алиса добавляет к своему сообщению аутентификатор* $t = (\underline{m})_3 + 1 \equiv 0 \pmod 2$. *Поэтому Алиса посылает*

```
i = 3; lam = 1;
{mes, Mod[m[[i]] + lam, 2]}
```

```
|| {7, 0}
```

Противник Ева, наблюдая это кодовое слово, может заключить, что ключ лежит в множестве $\{(i, \lambda) | 1 \leq i \leq 7, m_i + \lambda \equiv t \pmod 2\} = \{(1,0), (2,1), (3,1), (4,1), (5,0), (6,0), (7,1)\}$. *Чтобы проверить это, используем функции* Table *и* Mod *из "Mathematica".*

```
t = 0;
T = Table[{i, Mod[t - m[[i]], 2]}, {i, 1, 7}]
```

```
|| {{1, 0}, {2, 1}, {3, 1}, {4, 1}, {5, 0}, {6, 0}, {7, 1}}
```

Допустим, что Ева хочет послать сообщение 5. *Соответствующее кодовое слово* $\underline{m}'$ *задается так:*

```
mes' = 5;
inf' = IntegerDigits[mes', 2, 4];
m' = Mod[inf' * G, 2]
```

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

Если Ева выбирает в качестве аутентификатора $t' = 0$, то ее сообщение будет принято с вероятностью 4/7, потому что в точности четыре возможных ключа приводят к этому аутентификатору. С аутентификатором $t' = 1$ эта вероятность составит 3/7. (Мы используем функции Length и Intersection пакета "Mathematica".)

```
t' = 0;
T' = Table[{i, Mod[t' - m'[[i]], 2]}, {i, 1, 7}]
Length[Intersection[T, T']]
```

```
{{1, 0}, {2, 1}, {3, 0}, {4, 1}, {5, 0}, {6, 1}, {7, 0}}
4
```

Пример 13.10

q-арный код Рида–Соломона размерности k (см. [MacWS77]) имеет длину $n = q - 1$ и минимальное расстояние $d_H = n - k$. Умножая каждую координату на подходящую константу, можно предположить, что $\underline{1} \in C$. Теорема 13.10 *дает A-код с параметрами*

$$|\mathcal{M}| = q^{k-1}, \qquad |\mathcal{K}| = (q-1)q, \qquad |\mathcal{T}| = q;$$

$$P_I = 1/q, \qquad P_S \leq k/(q-1).$$

Конечно, метод, изложенный в этом подразделе, — не единственный способ получать A-коды из EC-кодов. Он обладает тем свойством, что каждая атака имперсонификации имеет одну и ту же вероятность успеха (именно, $1/q$).

Из того, что для любого сообщения каждый символ из $\mathcal{T} = \mathrm{GF}(q)$ может служить аутентификатором, вытекает, что множество C кодовых слов имеет мощность $|\mathcal{M}| \cdot q$. Это влечет, что в теореме 13.2 выполняется равенство.

В [JohKS93] авторы показывают также, как преобразовать A-код в код, исправляющий ошибки.

13.4 Задачи

Задача 13.1. Докажите, что из свойств PP-1, PP-2 и PP-3 в определении 13.5 следует, что проективная плоскость содержит четыре прямых, никакие три из которых не проходят через одну и ту же точку.

Задача 13.2. Докажите, что плоскость Фано единственна (с точностью до переименования точек и прямых).

Задача 13.3. Сравните конструкцию кода аутентификации, определяемого проективной плоскостью (см. определение 13.6) с кодом аутентификации, имеющим параметры $\mathcal{M} = \mathcal{K} = C = \mathbb{Z}_q$ и определяемым одноразовым щитом,

т.е. $m \mapsto c$, где $c \equiv m + k \pmod{q}$. Рассмотрите тот же вопрос, когда $\mathcal{M}$ — случайное подмножество из $\mathbb{Z}_q$ размера $\sqrt{q}$.

Задача 13.4. Проверьте, что строки в матрице инцидентности из примера 13.6 можно переставить так, чтобы новая матрица оказалась циркулянтом (т.е. каждая строка — циклический сдвиг вправо предыдущей строки).

Задача 13.5^M. Используйте ту же технику, что и в примере 13.6, чтобы определить верхнюю строку матрицы инцидентности проективной плоскости порядка 5. Циклически сдвигая эту строку, проверьте, что она определяет проективную плоскость порядка 5.

Задача 13.6^M. Преобразуйте ортогональный массив OA(4,5,1) из примера 13.7 в проективную плоскость порядка 4.

Задача 13.7. Покажите, что условие (13.11) в теореме 13.10 можно заменить требованием, чтобы C содержало по крайней мере одно кодовое слово веса[6] n.

Задача 13.8^M. Повторите пример 13.9 (обе части) для тернарного $(11,3^6,5)$-кода, порожденного матрицей

$$\mathtt{G} = \begin{pmatrix} 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 \end{pmatrix}.$$

[6] *Вес* (Хэмминга) вектора — это число его ненулевых координат, т.е. расстояние Хэмминга между данным вектором и нулевым вектором. — *Прим. ред.*

Глава 14

Протоколы с нулевым разглашением

Криптографический протокол — это обмен данными между двумя или более сторонами, следующий точным порядку и формату, с целью достижения особой безопасности. Разумеется, данное выше определение не слишком точное, но мы уже сталкивались с некоторыми примерами криптографических протоколов. Один из них — протокол проверки идентичности в разд. 4.1.2, другой — протокол Диффи–Хеллмана обмена ключами, еще несколько других упоминались в разд. 8.2.

Доказательство с нулевым разглашением (zero-knowledge proof) — это техника убеждения одним субъектом другого в том, что первый обладает определенными знаниями, но при этом не раскрывается ни один бит информации об этих знаниях. Как следствие, ни проверяющий, ни пассивный перехватчик не приобретают никакой информации от участия в любом числе итераций протокола.

Можно представить применение протокола с нулевым разглашением в ситуации, когда некто хочет использовать ATM[1], чтобы снять деньги с банковского счета. Вместо того, чтобы вводить PIN-код, достаточно убедить кассира в том, что этот некто знает PIN-код. Он хочет сделать это таким образом, чтобы никакая информация о PIN-коде не раскрывалась. В ближайшем разделе мы дадим пример того, как это может быть сделано. В разд. 14.2 представлена другая техника проверки идентичности.

14.1 Протокол Фиата–Шамира

Мы снова, как в подразд. 4.1.2, рассматриваем ситуацию, когда smart-карта хочет убедить устройство чтения карт в своей подлинности. Доверенная сторона, которая выпускает эти карты, выбирает большое составное число n, например, берет произведение двух больших простых чисел p и q, как в системе RSA. Число n — системный параметр, известный всем сторонам.

Безопасность *протокола Фиата–Шамира* [FiaS87] базируется на до-

[1] По-видимому, это — Asynchronous Transfer Mode (асинхронный режим передачи), базовая технология телекоммуникации, позволяющая передавать данные (включая изображение и звук) с очень высокой скоростью. — *Прим. ред.*

пущении, что вычисление квадратных корней по модулю большого составного числа n, вообще говоря, очень трудна. Это то же самое допущение, которое делалось при рассмотрении варианта Рабина системы RSA (разд. 9.5). В теореме 9.18 было показано, что проблема нахождения квадратного корня по модулю составного числа так же трудна, как и факторизация последнего.

Доверенная сторона вычисляет личный номер ID для smart-карты, который должен обладать дополнительным свойством:

$$ID \equiv s^2 \pmod{n} \tag{14.1}$$

для некоторого целого s. Номер ID может вычисляться, исходя из имени владельца карты и других относящихся к делу данных, но несколько битов нужно оставить открытыми для заполнения их доверенной стороной, чтобы сделать ID квадратом целого числа по модулю n (ID должен быть квадратичным вычетом по модулю p и по модулю q).

Доверенная сторона вычисляет квадратный корень s из ID (она может сделать это, так как знает разложение n; см. подразд. 9.5.3) и записывает его в сегмент памяти smart-карты, который недоступен внешнему миру.

Один раунд протокола Фиата–Шамира изображен ниже на рис. 14.1.

Рис. 14.1. Протокол идентификации Фиата–Шамира (один раунд).

Smart-карта или владелец карты сообщает устройству чтения карт личный номер ID. Для доказательства того, что карта действительно выпущена доверенной стороной, карта хочет убедить устройство чтения, что она знает s, квадратный корень из ID по модулю n.

С этой целью карта генерирует случайное число r, вычисляет его квадрат

$$t = r^2 \bmod n \tag{14.2}$$

и посылает устройству чтения. На жаргоне, принятом в этой области, t называется *свидетелем* знания картой числа r.

Устройство чтения выбирает случайное число e из $\{0, 1\}$ и предъявляет его карте в качестве *вызова* (или — на военном жаргоне — оклика). Как именно протокол *откликается* (отзывается) на вызов, зависит от значения e.

Если $e = 0$, то карта просто посылает случайное число r. Тогда устройство чтения проверяет, действительно ли его квадрат равен полученному ранее от карты значению t.

Если $e = 1$, то карта вычисляет $u = r \cdot s \bmod n$, произведение случайного числа r и секретного квадратного корня s, и посылает u устройству чтения карт. Устройство чтения проверяет, будет ли u^2 равно $t \times \mathrm{ID}$ по модулю n, что должно иметь место, поскольку $t = r^2 \bmod n$, а $\mathrm{ID} \equiv s^2 \pmod n$.

На рис. 14.1 эти две альтернативы объединены в отклике $u = r \cdot s^e \bmod n$. Устройство чтения проверяет, будет ли

$$u^2 \equiv t \cdot \mathrm{ID}^e \pmod n. \tag{14.3}$$

Должно быть ясно, что если карта может предоставить r (когда $e = 0$) и в то же время предоставить $r \cdot s$ (когда $e = 1$), то она должна знать квадратный корень s из ID. Также ясно, что если smart-карта не проходит тест (14.3), то устройство чтения должно отвергнуть эту smart-карту.

Если неправомочная smart-карта заблаговременно знает значение вызова e, то она может обмануть устройство чтения. Это очевидно в случае $e = 0$. В самом деле, тогда smart-карта берет случайное r, представляет $t = r^2 \bmod n$ в качестве свидетеля, а позже представляет само r в качестве отклика. В этих вычислениях секретный квадратный корень s никакой роли не играет.

Если нелигитимная карта знает, что вызовом будет 1, она порождает случайное r, вычисляет $t = (r^2/ID) \bmod n$ и представляет это значение t устройству чтения. После получения вызова $e = 1$ smart-карта представит $u = r$. Устройство чтения проверяет (см. (14.3)), будет ли u^2 сравнимо с $t \cdot \mathrm{ID}$ по модулю n. А это именно так в случае $u = r$ и $t \equiv r^2/\mathrm{ID} \pmod n$.

Заметим, что неправомочная карта не может правильно ответить на вызов, если ее догадка о вызове неверна. Поэтому smart-карта будет "поймана" с вероятностью $1/2$, если она случайно предсказывает вызов.

По этой причине smart-карта и устройство чтения карт выполняют данный выше протокол k раз, где k — параметр безопасности. Smart-карта, не знающая значения s, может угадать k случайных вызовов с вероятностью $(1/2)^k$, поэтому будет "поймана" с вероятностью $1 - (1/2)^k$.

Карта не должна использовать дважды одно и то же значение r, потому что как только устройство чтения знает как r, так и $r \cdot s$ (через u), оно может вычислить секретный квадратный корень s.

Идея доказательства тех или иных вещей без раскрытия какой-либо информации о них противна интуиции, но очень плодотворна. Область приложений доказательств с нулевым разглашением постоянно расширяется.

Примерами служат схемы электронного голосования, которые позволяют отдать голос анонимным путем. С другой стороны, голосующий будет пойман, когда попытается голосовать дважды. В этих схемах можно проверить, что в окончательном подсчете все голоса были учтены.

Другое приложение — система оплаты, которая позволяет вам снять деньги с вашего счета в цифровой форме и потратить их анонимно. Даже ваш собственный банк не сможет отследить вас. Однако, если вы попытаетесь потратить деньги дважды, ваша личность может быть установлена.

14.2 Протокол идентификации Шнорра

Шнорровский протокол проверки идентичности ([Schn91]) базируется на трудности проблемы дискретных логарифмов (табл. 8.1). Как и в схеме Диффи–Хеллмана, все участники разделяют некоторые параметры. Прежде всего, это некоторое поле $\mathrm{GF}(q)$ (которое равно $\mathbb{Z}_q$, если q — простое) и простой делитель p числа $q-1$. Возьмем примитивный элемент ω из $\mathrm{GF}(q)$ и элемент $\alpha = \omega^{(q-1)/p}$. Тогда α — примитивный корень p-й степени из единицы. Это означает, что все элементы $1, \alpha, \ldots, \alpha^{p-1}$ различны и что $\alpha^p = 1$.

Пример 14.1 (часть 1)

Пусть $p = 104729$ *и* $q = 8p + 1 = 837833$. *Возьмем* $\omega = 3$ *и* $\alpha = \omega^{(q-1)/p} = \omega^8 = 6561$. *Для проверки того, что* q *— простое и что* $\omega = 3$ *— примитивный элемент в* $\mathbb{Z}_q$ *(что делает* α *примитивным корнем* p*-й степени из единицы), используем функции* Prime, PrimeQ *из "Mathematica" и функцию* MultiplicativeOrder *(определенную в приложении D), которая вычисляет мультипликативный порядок элемента.*

```
MultiplicativeOrder[a_, n_] := If[GCD[[a, n] == 1,
        DIvisors[EulerPhi[n]] //.
    {x_, y_} -> If[PowerMod[a, x, n] == 1, x, {y}]];
```

```
p = Prime[10000]
q = 8 * p + 1
PrimeQ[q]
om = 3; MultiplicativeOrder[om, q]
al = om^8
```

```
|| 104729
```

```
|| 837833

|| True

|| 837832

|| 6561
```

Каждый доказывающий участник P выбирает случайный показатель x_P, вычисляет $y_P = \alpha^{x_P}$ и объявляет это значение публичным. Предполагается, что другие участники способны проверить, что y_P в самом деле является публичным параметром участника P. Это можно реализовать, если доверенный участник подписывает y_P или если публичные значения расклеиваются на надежной "доске объявлений". Если кто-то иной, проверяющий V, хочет убедиться, что y_P принадлежит P, то он делает это, проверяя, что P знает соответствующее x_P. Конечно, P не хочет раскрывать кому-либо секретное значение x_P. Поэтому он использует криптографический протокол, чтобы убедить V в том, что он знает x_P.

Пример 14.1 (часть 2)

Доказывающий P имеет личный номер $y_P = 693$ и секретный показатель $x_P = 18126$. Действительно, $\alpha^{18126} \equiv 693 \pmod{q}$.

```
xP = 18126; yP = 693;
PowerMod[al, xP, q] == yP
```

```
|| True
```

Рис. 14.2. Протокол Шнорра идентификации.

Протокол Шнорра идентификации действует следующим образом. Проверяющему предоставляется личный номер y_P участника P. Далес, участник P генерирует случайный показатель $r, 0 \leq r < p$, вычисляет

$\rho = \alpha^r$ и предоставляет это значение ρ проверяющему V как свидетеля своего секрета x_P. Проверяющий выбирает случайное число $s, 0 \leq s < p$, и вручает его P как вызов. Доказывающий P откликается вычислением $u = r + s \cdot x_P$ и дает V это значение. Проверяющий убеждается, что $\alpha^u = \rho \cdot (y_P)^s$. Это соотношение должно выполняться, поскольку $\alpha^u = \alpha^{r+s \cdot x_P} = \alpha^r \cdot (\alpha^{x_P})^s = \rho(y_P)^s$. Соответствующая схема изображена на рис. 14.2.

Пример 14.1 (часть 3)

Описанный выше протокол выполняется при указанном ниже входе. Используются функции Random, Mod и PowerMod из "Mathematica".

```
r = Randon[Integer, p]; rho = PowerMod[al, r, q];
Print["witness is ", r]
a = Random[Integer, p]; Print["challenge is ", s]
u = Mod[r + s * xP, p]; Print["response is ", u]
PowerMod[a, u, q] == Mod[rho * PowerMod[yP, s, q], q]
```

```
|| witness is 36431
|| challenge is 29041
|| response is 65643
|| True
```

Конечно, доказывающий должен дать только правильный отклик, если он знает x_P, удовлетворяющее равенству $\alpha^{x_P} = y_P$. Если он не знает x_P, он может догадаться о правильном значении u с вероятностью $1/p$. Значение p должно быть очень большим, чтобы сделать неподдающейся проблему дискретных логарифмов (см. подразд. 8.1.1).

Заметим, что в соотношении $u = r + s \cdot x_P$ проверяющему U известны только u и s. Другими словами, случайное значение r гарантирует, что никакая информация об x_P не просачивается к V. Это наблюдение показывает также, что доказывающий не должен использовать дважды одно и то же случайное значение r. В самом деле, из двух соотношений $u_1 = r + s_1 \cdot x_P$ и $u_2 = r + s_2 \cdot x_P$ с известными s_1, s_2, u_1, u_2 проверяющий легко может определить r и секретное x_P: мы получаем $x_P = (u_1 - u_2)/(s_1 - s_2)$.

Пример 14.1 (часть 4)

Для того же самого свидетеля генерируем вторые вызов и отклик:

```
ss = Random[Integer, p]; Print["second challenge is ", ss]
uu = Mod[r + ss * xP, p]; Print["second response is ", uu]
PowerMod[al, u, q] == Mod[rho * PowerMod[yP, s, q], q]
```

```
|| second challenge iz 62706
|| second response iz 21550
```

```
|| True
```

Далее вычисляем $x_P = (u_1 - u_2)/(s_1 - s_2)$*:*

```
Mod[(u - uu) * PowerMod[s - ss, -1, p], p]
```

```
|| 18126
```

Значение 18126 *действительно является секретным показателем* x_P *доказывающего.*

14.3 Задачи

Задача 14^M. Продублируйте пример 14.1 для $p = 113$. Найдите подходящее значение q.

Глава 15

Системы разделения секрета

15.1 Введение

В этой главе мы не будем вводить новой криптосистемы, а обсудим родственный вопрос. Начнем с примера из [Liu68].

> "Одиннадцать ученых работают над секретным проектом. Им нужно хранить документы в кабинете. Кабинет должен открываться в присутствии не менее, чем шести ученых. Каково минимальное число необходимых для этого замков? Какое минимальное число ключей должен носить с собой каждый из ученых?"

Ясно, что для каждых пяти ученых должен существовать замок, который никто из них не может открыть. При этом у каждого из шести оставшихся ученых должен быть ключ от этого замка. Более одного такого замка для пятерки не требуется. Итак, требуется $\binom{11}{5}$ замков, и каждый ученый должен носить $\binom{11-1}{5}$ ключей. Эти числа можно вычислить с помощью функции *Binomial*, пакета "Mathematica".

```
Binomial[11, 5]
Binomial[11 - 1, 5]
```

```
462
```

```
252
```

Конечно, полученное решение нельзя считать удобным с практической точки зрения. Но и описанная ситуация выглядит не очень реалистичной. Однако, существуют вполне реальные ситуации, в которых требуется разделить некоторую важную информацию в группе людей таким образом, что только определенные привилегированные группы должны иметь возможность открыть секретную информацию. Примерами являются мастер-ключ или частный ключ системы платежей, которые нежелательно хранить в одном месте.

В любой такой ситуации, если P — привилегированная группа участников в том смысле, что эти участники вместе имеют право доступа к секрету, то каждая группа, частью которой является P, тоже должна быть привилегированной. А также каждая часть непривилегированной группы N тоже не будет привилегированной.

Определение 15.1
Структура доступа $(U, \mathcal{P}, \mathcal{N})$ состоит из множества пользователей U и двух непересекающихся наборов его подмножеств $\mathcal{P}$ и $\mathcal{N}$ ($\mathcal{P}$ — *привилегированные*, а $\mathcal{N}$ — *непривилегированные*), обладающих теми свойствами, что

$$\begin{aligned} P \in \mathcal{P},\ P \subset B \subset U &\implies B \in \mathcal{P}, \\ N \in \mathcal{N},\ A \subset N &\implies A \in \mathcal{N}. \end{aligned}$$

В приведенном выше примере $U = \{1, 2, ..., 11\}$, $\mathcal{P}$ состоит из всех подмножеств в U размера не менее 6, а $\mathcal{N}$ — из все остальных подмножеств в U. Это особый случай, который обобщенно называется *пороговой схемой*. Часто оказывается удобным вносить в список лишь подмножество *минимальных элементов* множества $\mathcal{P}$, обозначаемое $\mathcal{P}^-$, которое можно получить, удалив из $\mathcal{P}$ каждый элемент, строго содержащий в себе какой-нибудь другой элемент из $\mathcal{P}$. Подобным же образом $\mathcal{N}$ часто представляют с помощью подмножества $\mathcal{N}^+$ *максимальных элементов*.

Структура доступа называется *полной* или *совершенной*, если каждое подмножество множества U лежит либо в $\mathcal{P}$, либо в $\mathcal{N}$.

Определение 15.2
Пусть S — случайная величина, равномерно распределенная на множестве $\mathbb{S}$.
Пусть U — группа из n участников, каждый из которых получил отдельный элемент S_i, зависящий от элементов из $\mathbb{S}$ от лица, пользующегося общим доверием. Пусть теперь $(U, \mathcal{P}, \mathcal{N})$ — структура доступа. Тогда набор $\{S_i\}_{i \in U}$ называется *схемой разделения секрета* для $(U, \mathcal{P}, \mathcal{N})$, если он обладает двумя следующими свойствами:

СРС1: каждая привилегированная группа P участников ($P \in \mathcal{P}$) способна вычислить секрет S.

СРС2: каждая непривилегированная группа N участников ($N \in \mathcal{N}$) не способна получить никакой информации об S.

Величину S_i (называемую i-й *долей*) можно интерпретировать как частичную информацию i-го участника о секрете S. В терминах теории информации (см. гл. 5), СРС1 и СРС2 можно переформулировать так:

СРС1: $H(S \mid \{S_i\}_{i \in P}) = 0$ для любого $P \in \mathcal{P}$,

СРС2: $H(S \mid \{S_i\}_{i \in N}) = H(S)$ для любого $N \in \mathcal{N}$.

Отметим, что в несовершенных схемах разделения секрета могут быть такие коалиции C ($C \notin \mathcal{P} \cup \mathcal{N}$) участников, которые способны получить некоторую информацию о секрете S (поскольку $H(S \mid \{S_i\}_{i \in C}) < H(S)$), не будучи привилегированными.

15.2 Пороговые схемы

Схема разделения секрета $\{S_i\}_{i=1}^n$ называется (n,k)–пороговой схемой, если $\mathcal{P}$ состоит из всех подмножеств в U мощности $\geq k$, а $\mathcal{N}$ состоит из всех подмножеств в U мощности $\leq k-1$. Пороговая схема по определению является совершенной схемой разделения секрета. Свойства СРС1 и СРС2 для нее можно переформулировать так:

ПС1: Знание k и более различных S_i позволяют вычислить S.

ПС2: Знание не более $k-1$ различных S_i оставляют секрет S совершенно неопределенным, а точнее, оставляют все возможные значения из $\mathbb{S}$ одинаково правдоподобными.

Шамир описал (см. [Sham79]) следующую общую конструкцию (n,k)-пороговых схем, в которых $\mathbb{S}$ — конечное поле $GF(q)$, где q должно быть больше, чем n. Мы будем здесь предполагать, что число q — простое, скажем, $q=p$, в таком случае $\mathbb{S}$ — это просто $\mathbb{Z}_p$, т.е. множество целых чисел по модулю p. Обобщение этого случая на произвольное поле $GF(q)$ выполняется непосредственно.

Конструкция 15.1
Пусть участники помечены числами от 1 до n, и пусть $S \in \mathbb{Z}_p$, $p > n$, — секретные данные. Рассмотрим многочлен

$$f(x) = S + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}, \qquad (15.1)$$

степени не выше $k-1$, коэффициенты a_j, $1 \leq j \leq k-1$, которого случайным образом выбираются из $\mathbb{Z}_p$ лицом, пользующимся общим доверием (*арбитром*). Участнику i, $1 \leq i \leq n$, выдастся в качестве его доли S_i пара

$$S_i = (i, f(i) \bmod p). \qquad (15.2)$$

Пример 15.1 (часть 1)

Для того, чтобы сконструировать $(10,4)$*–пороговую схему для секрета* $S = 17$ *из* $\mathbb{Z}_{19}$, *мы скроем секрет в многочлене* $f(x)$ *(отметим использование функции* Mod *пакета "Mathematica"),*

```
Clear[f];
f[x_] := Mod[17 + 7x + 12x^2 + 5x^3, 19]
```

в котором коэффициенты при x^j, $1 \leq j \leq 3$, *выбраны случайно из* $\mathbb{Z}_{19}$.

Значения долей можно вычислить с помощью функции Table *пакета "Mathematica".*

```
Table[{i, f[i]}, {i, 1, 10}]
```

```
{{1, 3}, {2, 5}, {3, 15}, {4, 6}, {5, 8},
{6, 13}, {7, 13}, {8, 0}, {9, 4}, {10, 17}}
```

Чтобы убедиться в том, что величины S_i, $1 \le i \le n$, заданные формулой (15.2), образуют (n,k)–пороговую схему, мы должны проверить два условия ПС1 и ПС2.

Проверка ПС1.

Предположим, что участники $i_1, i_2, ..., i_k$ объединяют свои доли $S_{i_1} = (i_1, f(i_1))$, $S_{i_2} = (i_2, f(i_2))$, ..., $S_{i_k} = (i_k, f(i_k))$. С помощью *интерполяционной формулы Лагранжа* легко определить $f(x)$. Действительно,

$$f(x) = \sum_{u=1}^{k} f(i_u) \prod_{\ell=1, \ell \neq u}^{k} \frac{x - i_\ell}{i_u - i_\ell}. \qquad (15.3)$$

Поскольку степень выражения в правой части (она равна $k-1$) совпадает со степенью $f(x)$ (см. (15.1)), и поскольку значения этого выражения при $x = i_j$, $1 \le j \le k$, совпадают со значениями $S_j = f(i_j)$, многочлен в правой части совпадает с $f(x)$. Отметим, что по (15.1) секрет S определяется как $f(0)$, следовательно, достаточно взять $x = 0$ при вычислении по интерполяционной формуле Лагранжа.

Пример 15.1 (часть 2)

Предположим, что участники 1, 3, 6 *и* 9 *хотят восстановить секрет* S. *Они объединяют свои доли* $(1,3)$, $(3,15)$, $(6,13)$ *и* $(9,4)$.

Вычислить интерполяционный многочлен Лагранжа можно с помощью функции InterpolatingPolynomial *пакета "Mathematica". Функция* PolynomialMod *используется для вычисления коэффициентов по модулю* 19.

```
PolynomialMod[InterpolatingPolynomial[
{{1, 3}, {3, 15}, {6, 13}, {9, 4}}, x], 19]
```

$$17 + 7x + 12x^2 + 5x^3$$

Секретная величина S *— это свободный член полученного многочлена, т.е.* $S = 17$.

Проверка ПС2.

Допустим, что известны доли S_{i_1}, S_{i_2}, ..., S_{i_ℓ} для некоторого $\ell < k$. Из соотношений (15.1) и (15.3) следует, что ровно $q^{k-\ell-1}$ многочленов $g(x)$, имеющих произвольное наперед заданное значение $g(0)$, удовлетворяют условию $g(i_u) = S_{i_u}$, $1 \le u \le \ell$.

Действительно, для каждого зафиксированного значения $g(0)$ можно присоединить к исходной группе $k - \ell - 1$ любых дополнительных участников, произвольно выбрав их доли. Из интерполяционной формулы Лагранжа следует, что таким требованиям будет удовлетворять ровно один многочлен $g(x)$.

Пример 15.1 (часть 3)

Предположим, что участники 1, 3 *и* 9 *пытаются восстановить секрет* S, *объединив свои доли* $(1,3)$, $(3,15)$ *и* $(9,4)$.

В такой ситуации секрет S *по-прежнему может принимать любое значение (и все эти значения по-прежнему одинаково правдоподобны). Действительно, для любой пары* $(0,S)$ *существует ровно один многочлен* $g(x)$, *удовлетворяющий условиям* $g(0)=S$, $g(1)=3$, $g(3)=15$ *и* $g(9)=4$. *Это следует из интерполяционной формулы Лагранжа и проверяется следующим образом.*

```
Clear[x]
Table[ {S, PolynomialMod[InterpolatingPolynomial[
    {{0, S}, {1, 3}, {3, 15}, {9, 4}}, x], 19]},
        {S, 0, 18} ] // Tableform
```

```
0    2x + x^2
1    1 + 9x + 5x^2 + 7x^3
2    2 + 16x + 9x^2 + 14x^3
3    3 + 4x + 13x^2 + 2x^3
4    4 + 11x + 17x^2 + 9x^3
5    5 + 18x + 2x^2 + 16x^3
6    6 + 6x + 6x^2 + 4x^3
7    7 + 13x + 10x^2 + 11x^3
8    8 + x + 14x^2 + 18x^3
9    9 + 8x + 18x^2 + 6x^3
10   10 + 15x + 3x^2 + 13x^3
11   11 + 3x + 7x^2 + x^3
12   12 + 10x + 11x^2 + 8x^3
13   13 + 17x + 15x^2 + 15x^3
14   14 + 5x + 3x^3
15   15 + 12x + 4x^2 + 10x^3
16   16 + 8x^2 + 17x^3
17   17 + 7x + 12x^2 + 5x^3
18   18 + 14x + 16x^2 + 12x^3
```

Замечание 1. При обобщении на произвольное поле $GF(q)$ n участников помечены различными ненулевыми элементами поля a_i, $1 \le i \le n$. Долей S_i i-го участника будет пара $(a_i, f(a_i))$. Это можно реализовать, выбрав в поле $GF(q)$ примитивный (порождающий) элемент α, пометив участников числами от 1 до n, и выдав i-му участнику в качестве его доли пару $(i, f(\alpha^i))$.

Замечание 2. Представленная здесь пороговая схема требует участия арбитра. Кроме того, эта схема может использоваться лишь один раз. Как только участники обменялись своими долями для получения секрета, эти доли стали не секретны. Для дальнейшего использования

схемы нужно распределить новые доли. В литературе можно найти предложения, позволяющие ослабить эти условия.

15.3 Пороговые схемы с лгунами

В [McEl81] предложен вариант описанной выше конструкции, в котором некоторые из участников представляют ложную информацию, т.е. представленные ими доли содержат неправильные значения. Некоторые участники могут пойти на это с целью помешать другим получить доступ к секретным данным. Окажется так, что для раскрытия секрета на каждую ложную долю потребуется добавить по две дополнительных доли. Итак, если $k+2t$ участников объединяют свои доли для раскрытия секрета, то среди этих долей должно быть не более t ложных.

Конструкция 15.2
Пусть S — секретный элемент поля $GF(q)$, где q — некоторая степень простого числа, и пусть $\alpha_1, \alpha_2, ..., \alpha_n$, $n \leq q-1$, — список из n различных ненулевых элементов $GF(q)$, например, $\alpha_i = \alpha^i$, $1 \leq i \leq n$, для некоторого примитивного элемента $\alpha \in GF(q)$.
Рассмотрим многочлен $f(x) = S + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1}$, коэффициенты a_j, $1 \leq j \leq k-1$, которого случайным образом выбираются из $GF(q)$. Пара $(\alpha_i, f(\alpha_i))$ будет долей S_i i-го участника.
Предположим, что $k+2t$ участников ($k+2t \leq n$) объединяют свои доли для раскрытия секрета и среди этих долей не более t неправильных. Тогда объединившиеся участники смогут эффективно вычислить $f(x)$ и раскрыть секрет S. Более того, при этом можно распознать и неправильные доли.

Доказательство. Используемый при вычислении долей многочлен $f(x)$ имеет степень $\leq k-1$ и обладает тем дополнительным свойством, что на нем лежат (как точки на графике функции $y = f(x)$) по крайней мере $k+t$ правильных долей. Может ли быть какой-то другой многочлен, скажем, $g(x)$, обладающий теми же свойствами? Ответ — нет. Действительно, в $(k+2t)$-элементном множестве долей любые два не менее, чем $(k+t)$-элементных подмножества правильных долей, должны иметь пересечение, состоящее не менее, чем из k элементов (правильных долей). Эти k долей лежат как на $f(x)$, так и на $g(x)$. Поскольку степени обоих многочленов не превосходят $k-1$, из этого следует, что $f(x) = g(x)$.

Чтобы найти $f(x)$, участники должны перепробовать все многочлены степени $\leq k-1$, проходящие через k долей, пока не найдется многочлен, проходящий через $k+t$ или более долей. Конечно, такой способ не эффективен. Для эффективного поиска требуется использование теории кодов, исправляющих ошибки (как в гл. 11). Определенные выше доли в действительности определяют кодовые слова $(f(\alpha_1), f(\alpha_2), ..., f(\alpha_n))$ так

называемого укороченного кода Рида-Соломона с параметрами $(n, k, n-k+1)$.

Читатель, не знакомый с этой теорией, может обратиться к гл. 11 [MacWS77]. Эффективными способами декодирования этого кода являются алгоритм Берлекэмпа-Масси и алгоритм Эвклида. В контексте нашей проблемы, где известны $k+2t$ долей, можно считать остальные $n-k-2t$ долей стертыми. Если число стертых долей плюс удвоенное число ошибок меньше, чем минимальное расстояние кода, то код исправит эти стертые символы и ошибки. У нас $(n-k-2t)+2\cdot t$, действительно, меньше, чем $n-k+1$. Для кодов Рида-Соломона существует эффективный алгоритм декодирования с исправлением этих ошибок и восстановлением стертых символов (см. [Berl68], разд. 10.4, и [SugK76]). ∎

Замечание 1. При $t=0$ конструкция 15.2 сводится к конструкции 15.1.

Замечание 2. Если доступно лишь $k+2t-1$ долей и t из них — неправильные, то $f(x)$ не всегда определяется однозначно. Например, возможно, что из $k+2t-1$ имеющихся долей, все, кроме t первых, лежат на одном многочлене степени $k-1$, и, одновременно, все, кроме t последних долей, лежат на другом многочлене степени $\leq k-1$ (пересечение множеств долей имеет мощность $k-1$).

Однако, в такой ситуации некоторая информация о секрете становится известной.

Пример 15.2

Рассмотрим $k=3$, $t=1$ и $p=17$. Любые три из четырех долей $(1,4)$, $(2,1)$, $(3,5)$, $(4,4)$ определяют параболу, на которую четвертая точка не попадает, и поэтому может считаться неправильной.

```
PolynomialMod[InterpolatingPolynomial[
    {{1, 4}, {2, 1}, {3, 5}}, x], 17]
PolynomialMod[InterpolatingPolynomial[
    {{1, 4}, {2, 1}, {4, 4}}, x], 17]
PolynomialMod[InterpolatingPolynomial[
    {{1, 4}, {3, 5}, {4, 4}}, x], 17]
PolynomialMod[InterpolatingPolynomial[
    {{2, 1}, {3, 5}, {4, 4}}, x], 17]
```

```
|| 14 + 12x + 12x^2
|| 10 + x + 10x^2
|| 2 + 11x + 8x^2
|| 12 + 8x + 6x^2
```

Из 17 возможных секретов в описанной ситуации остаются возможными четыре, причем все они равновероятны.

15.4 Схемы разделения секрета

Несмотря на большой объем литературы по схемам разделения секрета, много центральных вопросов в этой области остаются без ответов. Поэтому мы обсудим лишь один пример схемы разделения секрета. Чтобы найти обсуждение различных обобщений описанной здесь техники, читатель может обратиться к [Bric89] и [Dijk97]. Общее введение в схемы разделения секрета можно найти в [Stin95].

Допустим, что у нас есть структура доступа $(U,\ \mathcal{P},\ \mathcal{N})$, в которой $U = \{1,2,3,4\}$, $\mathcal{P}^- = \{\{1,2\},\{2,3\},\{3,4\}\}$ и $\mathcal{N}^+ = \{\{1,3\},\{1,4\},\{2,4\}\}$. Это означает, что любое подмножество U, включающее в себя какую-то из пар пользователей 1 и 2, 2 и 3, или 3 и 4, является привилегированным, а любая другая группа пользователей является непривилегированной. Эта ситуация изображена на рис. 15.1.

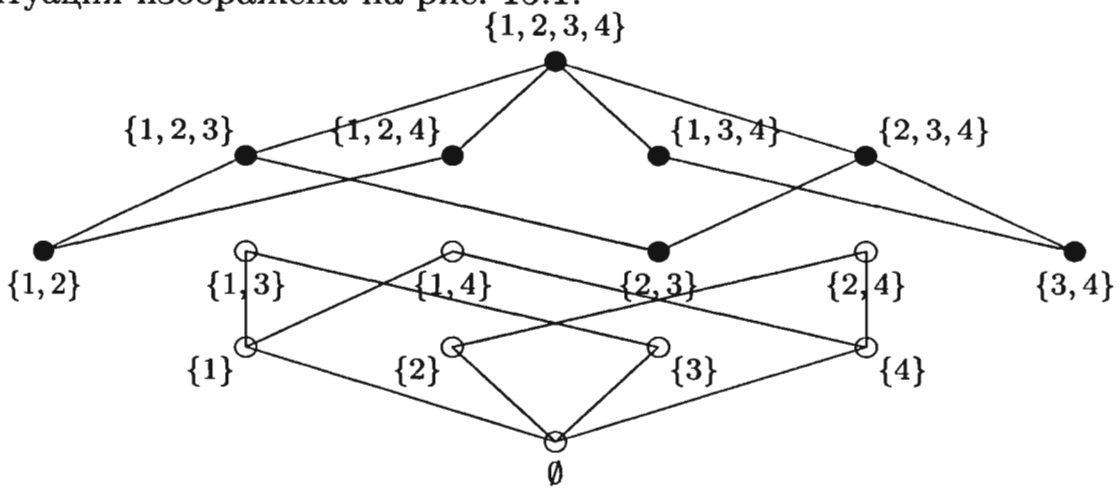


Рис. 15.1. Структура доступа с четырьмя участниками; • — привилегированные, ○ — непривилегированные.

Схема разделения секрета для данной структуры доступа строится за два шага. На первом шаге мы хотим разделить один бит (байт, строку) секретной информации среди четырех участников.

Пусть s — секретный бит, который нужно разделить среди участников нашей структуры $(U,\ \mathcal{P},\ \mathcal{N})$.

Арбитр выбирает два случайных бита a и b и выдаст участникам следующие доли:

УЧАСТНИК	ДОЛЯ
1	a
2	$s + a,\ b$
3	$s + b$
4	b

Рис. 15.2. Схема разделения секрета размером в один бит.

Знаком + обозначено сложение по модулю 2. Читатель легко может убедиться в том, что эта схема удовлетворяет требованиям CPC1 и CPC2.

Скажем, участники 1 и 2 могут вычислить s по формуле $a + (s + a)$, где a взято у участника 1, а $s + a$ — у участника 2.

Пример 15.3

Например, если арбитр хочет разделить секрет $s = 1$ среди четырех участников, то он может выбрать $a = 1$ и $b = 0$. Тогда долями участников 1, 2, 3 и 4 будут, соответственно, 1, $(0, 0)$, 1 и 0.

Участники 2 и 4 не могут раскрыть секрет, поскольку они вместе знают только $s + a$ и b (в двух экземплярах). Участники 3 и 4 могут раскрыть секрет, сложив свои доли $s + b$ и b: $1 + 0 = 1$.

Видно, что в схеме на рис. 15.2 участник номер 2 должен хранить долю вдвое большей, чем секрет, длины (в битах). Это отношение можно улучшить, если скомбинировать описанную схему с ее переставленной версией.

Итак, рассмотрим теперь секрет, состоящий из двух битов s_1 и s_2. Арбитр выбирает четыре случайных бита a, b, c и d. Затем он раздает участникам следующие доли:

УЧАСТНИК	ДОЛЯ
1	a, c
2	$s_1 + a$, $s_2 + c$, b
3	$s_1 + b$, $s_2 + d$, c
4	b, d

Рис. 15.3. Схема разделения секрета размером в два бита.

В этой схеме отношение размера секрета к размеру самой длинной доли равно 2/3 (это отношение называется *информационной эффективностью*). Можно показать, что такое отношение всегда не больше 1. Схемы разделения секрета с информационной эффективностью, равной 1, называются *идеальными*.

Для конструкций описанного выше типа есть матричное описание. Мы приведем его для построенного выше примера.

Схема разделения секрета описывается матрицей G_{TA} доверенного арбитра[1] и матрицами G_i участников 1, 2, 3 и 4. Первые два столбца всех этих матриц помечены секретными битами s_1 и s_2, а следующие четыре столбца — случайными переменными a, b, c и d. Каждая строка матрицы G_i выражает один элемент доли i-го участника (в виде линейной комбинации секретных битов и случайных битов), то же можно сказать и о матрице G_{TA}, если считать, что доля арбитра — это s_1 и s_2.

[1] TA от "Trusted Authority". — *Прим. ред.*

```
GTA = ( 1 0 0 0 0 0 )
      ( 0 1 0 0 0 0 );

Gp1 = ( 0 0 1 0 0 0 )
      ( 0 0 0 0 1 0 );

Gp2 = ( 1 0 1 0 0 0 )
      ( 0 1 0 0 1 0 )
      ( 0 0 0 1 0 0 );

Gp3 = ( 1 0 0 1 0 0 )
      ( 0 1 0 0 0 1 )
      ( 0 0 0 0 1 0 );

Gp4 = ( 0 0 0 1 0 0 )
      ( 0 0 0 0 0 1 );
```

Чтобы увидеть, что эти матрицы действительно представляют схему разделения секрета, умножим их на вектор (s_1, s_2, a, b, c, d).

```
Clear[s1, s2, a, b, c, d];
vec = {s1, s2, a, b, c, d};
GTA . vec
Gp1 . vec
Gp2 . vec
Gp3 . vec
Gp4 . vec
```

```
|| {s1, s2}

|| {a, c}

|| {a + s1, c + s2, b}

|| {b + s1, d + s2, c}

|| {b, d}
```

Мы получаем секрет арбитра и доли всех участников, что в точности составляет схему, которая была у нас ранее.

Теперь можно следующим образом переформулировать свойства схемы разделения секрета.

Теорема 15.3
Матрицы G_{TA} и G_i, $i \in U$, с линейно независимыми строками описывают схему разделения секрета для структуры доступа $(U, \mathcal{P}, \mathcal{N})$ тогда и только тогда, когда

i) для каждого привилегированного множества $A \in \mathcal{P}$ каждая строка матрицы G_{TA} лежит в линейной оболочке строк матриц G_i, $i \in A$,

ii) для каждого непривилегированного множества $B \in \mathcal{N}$ ни одна строка матрицы G_{TA} не лежит в линейной оболочке строк матриц G_i, $i \in B$.

Для того, чтобы проверить, что первая строка матрицы G_{TA} лежит в линейной оболочке строк матриц G_1 и G_2, используем пакет *LinearAlgebra'MatrixManipulation'* и функции *AppendColumns*, *MatrixForm*, *LinearSolve* и *Transpose*.

```
<< LinearAlgebra'MatrixManipulation';
```

```
u = GTA[[1]]
M = AppendColumns[Gp1, Gp2];
MatrixForm[M]
LinearSolve[Transpose[M], u, Modulus -> 2]
```

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

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

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

Это показывает, что первая строка матрицы G_{TA} является суммой по модулю 2 первой строки матрицы G_1 и первой строки матрицы G_2.

Аналогично можно проверить, что участники 1 и 3 не смогут определить s_2: вторая строка (и первая тоже) матрицы G_{TA} не лежит в линейной оболочке строк матриц G_1 и G_3.

```
u = GTA[[2]]
M = AppendColumns[Gp1, Gp3];
MatrixForm[M]
LinearSolve[Transpose[M], u, Modulus -> 2]
```

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

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

```
LinearSolve :: nosol :
Linear equation encountered which has no solution
```

```
LinearSolve[
{{0, 0, 1, 0, 0}, {0, 0, 0, 1, 0}, {1, 0, 0, 0, 0},
{0, 0, 1, 0, 0}, {0, 1, 0, 0, 1}, {0, 0, 0, 1, 0}},
{0, 1, 0, 0, 0, 0}, Modulus -> 2]
```

Закончим этот раздел замечанием о том, что проблема создания совершенной схемы разделения секрета не так сложна, как проблема сделать эту систему высоко информационно эффективной. Действительно, неэффективная схема разделения секрета для конкретной структуры доступа $(U,\ \mathcal{P},\ \mathcal{N})$ создается следующим образом.

Пусть требуется разделить секрет s. Для каждого $A \in \mathcal{P}^-$ выберем случайные биты $a_i^{(A)}$, $1 \le i \le |A|$, удовлетворяющие сравнению

$$\sum_{i=1}^{|A|} a_i^{(A)} \equiv s \pmod 2,\ A \in \mathcal{P}^-.$$

Каждый участник $u \in A$ получает один из этих $a_i^{(A)}$.

В примере $U = \{1,2,3,4\}$, $\mathcal{P}^- = \{\{1,2\},\{2,3\},\{3,4\}\}$ и $\mathcal{N}^+ = \{\{1,3\},\{1,4\},\{2,4\}\}$, таким способом получаем следующее разделение секрета s[2]:

УЧАСТНИК	ДОЛЯ
1	$a_1^{\{1,2\}}$
2	$s + a_1^{\{1,2\}},\ a_1^{\{2,3\}}$
3	$s + a_1^{\{2,3\}},\ a_1^{\{3,4\}}$
4	$s + a_1^{\{3,4\}}$

Вот более краткое представление схемы разделения секрета:

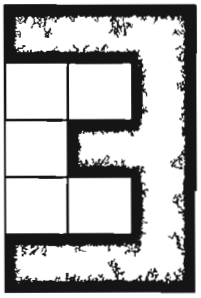
УЧАСТНИК	ДОЛЯ
1	a
2	$s + a,\ b$
3	$s + b,\ c$
4	$s + c$

Информационная эффективность этой схемы равна 1/2. В ней используются три случайные переменные в отличие от схемы, изображенной на рис. 15.2.

15.5 Визуальные схемы разделения секрета

В *визуальных схемах разделения секрета* секрет, который требуется разделить, представляет собой картинку, состоящую из белых и черных (или цветных) точек. Здесь мы обсудим только случай черно-белого изображения, при этом "белый" будем понимать как "прозрачный". Например, число 3 можно нарисовать следующим образом:

[2] $s + a_1^{(A)} = a_2^{(A)}$. — *Прим. перев.*



3

Доли представляют собой транспаранты той же формы, заполненные белыми и черными точками. Суть визуальной схемы разделения секрета для структуры доступа $(U, \mathcal{P}, \mathcal{N})$ состоит в том, что каждая привилегированная группа участников может получить секретное изображение, просто складывая свои транспаранты в стопку один на другой. С другой стороны, никакая непривилегированная группа участников не может с помощью своих долей получить никакой информации о секретном изображении.

Визуальную схему разделения секрета нельзя реализовать напрямую. В самом деле, если в такой реализации в какой-то доле некоторая точка — черная, то и в секретном изображении соответствующая точка окажется черной. Чтобы решить эту проблему, каждую точку в секрете и в долях делят на m подточек. Число m называется *расширяющим множителем* схемы. Предполагается, что при этом существуют два таких *пороговых визуальных значения* α, β, $0 \leq \alpha < \beta \leq 1$, что

- если не более, чем $\alpha \cdot m$ подточек черные, то вся точка воспринимается человеческим глазом как белая,

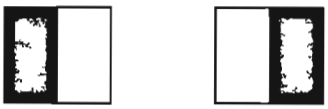
- если не менее, чем $\beta \cdot m$ подточек черные, то вся точка воспринимается как черная.

Если же число черных подточек лежит строго между $\alpha \cdot m$ и $\beta \cdot m$, то предполагается, что человеческий глаз не может решить, какого цвета точка. Разность $\beta - \alpha$ является индикатором уровня контрастности в изображении, все точки которого удовлетворяют одному из двух приведенных выше условий. Такое предположение обосновывается биологически, как необходимая глазу разница в интенсивности света. Дальнейшее обсуждение этого можно найти в [VerT97].

При построении визуальных схем разделения секрета перед нами стоят дополнительные проблемы. Например, если при наложении друг на друга долей из непривилегированного множества окажется, что некоторая точка содержит более, чем $\alpha \cdot m$ черных подточек, то мы узнаем, что в секретном изображении на этом месте черная точка. Разумеется, следует избегать таких ситуаций.

Ясно, что, имея визуальную схему разделения секрета для одной точки, можно ее распространить и на другие точки, создавая тем самым визуальную схему разделения секрета для целого изображения.

Здесь мы объясним лишь устройство визуальной схемы разделения секрета для $(n, 2)$-пороговой схемы. Это означает, что любые два участника вместе должны иметь доступ к секрету, а в одиночку никто не имеет информации даже об одной точке секретного изображения. Перед тем как приступить к общему случаю, рассмотрим простую ситуацию с двумя участниками. Выберем расширяющий множитель $m = 2$. Обозначим следующие две части деления точки через L и R (в соответствии с расположением в них черной подточки).

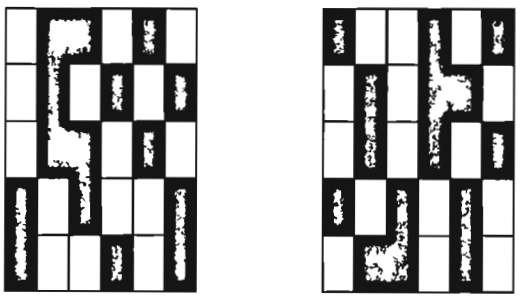


L R

Ясно, что при наложении L и R друг на друга получается черная точка, тогда как $L + L$ и $R + R$ дают точку, состоящую из двух подточек разного цвета. Таким образом, нам удалось изготовить конструкцию с пороговыми значениями $\alpha = 1/2$ и $\beta = 1$.

Конструкция 15.4
Чтобы разделить белую точку, арбитр выдает участникам одинаковые доли, при этом каждый участник может с равными вероятностями получить как L, так и R.
Чтобы разделить черную точку, арбитр выдает участникам разные доли, при этом вновь каждый участник может с равными вероятностями получить как L, так и R. Это дает визуальную $(2, 2)$-пороговую схему с расширяющим множителем $m = 2$ и пороговыми значениями $\alpha = 1/2$ и $\beta = 1$.

Ниже приведен пример возможных долей участников 1 и 2, если секретом было приведенное ранее изображение числа 3.



Доля 1 *Доля 2*

Читатель может проверить это, изготовив транспаранты таких форм и наложив их друг на друга.

Известно много конструкций для визуальных (n, k)-пороговых схем.

Опишем общую конструкцию для $k = 2$. В каждом конкретном случае будут получаться свой расширяющий множитель m и свои пороговые значения α и β. При этом возникают две $(n \times m)$-матрицы M_W и M_B,

используемые при распределении среди участников долей белой и, соответственно, черной точек. Эти матрицы в дальнейшем характеризуются двумя параметрами r и λ и должны обладать следующими свойствами:

ВПС1: Матрица M_W состоит из n одинаковых копий строки $\overbrace{11...1}^{r}\overbrace{00...00}^{m-r}$.

ВПС2: Сумма элементов в каждой строке матрицы M_B равна r.

ВПС3: Скалярное произведение каждой пары строк матрицы M_B равно λ.

Значения m, α, β, r и λ будут взаимосвязаны. Их нельзя выбирать произвольно.

Пример 15.4 (часть 1)

Возьмем $n = 4$ и $m = 6$. Пусть матрицы M_W и M_B задаются следующим образом.

$$\texttt{MW} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix};$$

$$\texttt{MB} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix};$$

Отметим, что M_W и M_B обладают свойствами ВПС1–ВПС3 для $r = 3$ и $\lambda = 1$.

Матрицы M_W и M_B определяют два класса $n \times m$-матриц:

$$\mathcal{M}_W = \{M_W \cdot P \mid P \text{ — матрица перестановки размера } m \times m\},$$

$$\mathcal{M}_B = \{M_B \cdot P \mid P \text{ — матрица перестановки размера } m \times m\}.$$

При распределении долей для конкретной секретной точки арбитр берет либо M_W, либо M_B в зависимости от того, белая эта точка или черная, переставляет столбцы случайным образом и выдает i-ю строку участнику с номером i, $1 \le i \le n$.

Участник номер j определяет цвет j-й подточки в зависимости от значения j-й компоненты своей доли. Белый цвет соответствует 0, а черный — 1.

Пример 15.4 (часть 2)

Предположим, что требуется разделить черную точку. Арбитр выбирает случайную перестановку с помощью функции RandomPermutation из пакета DiscreteMath'Permutations' следующим образом.

```
<< DiscreteMath`Permutations`
```

```
RP = RandomPermutation[6]
```

```
{3, 6, 4, 2, 1, 5}
```

Это приводит к следующей матрице перестановки (используются функции Table, Do и MatrixForm):

```
P = Table[0, {i, 1, 6}, {j, 1, 6}];
Do[P[[j, RP[[j]]]] = 1, {j, 1, 6}];
MatrixForm[P]
```

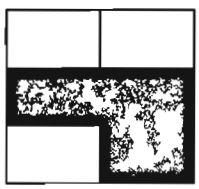
$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Умножение M_B на P справа дает матрицу

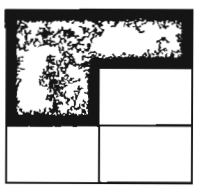
```
PMB = MB * P;
MatrixForm[PMB]
```

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

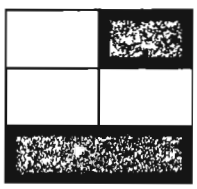
Располагая шестерки подточек по строкам массивов 3×2, получим четыре доли для черной точки:



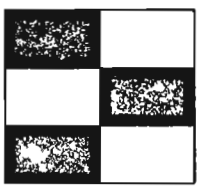
Доля 1



Доля 2



Доля 3



Доля 4

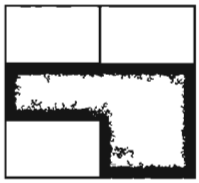
Читатель легко проверит, что при наложении друг на друга любых двух из этих картинок получатся пять черных подточек и одна белая.

Если исходная точка была белой, то получим

```
PMW = MW * P;
MatrixForm[PMW]
```

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

Это означает, что все четыре доли выглядят так:



Каждая доля

Строки матриц M_W и M_B содержат одинаковое количество единиц (а именно, по r), и $\mathcal{M}_W$ и $\mathcal{M}_B$ получены из них умножением справа на все возможные матрицы перестановок. Из этого следует, что каждый вектор длины m и веса r может с одинаковой вероятностью оказаться как долей для белой точки, так и долей для черной. Тем самым показано, что у нашей конструкции нижнее визуальное пороговое значение α равно r/m.

Поскольку M_W умножается на матрицу перестановки, из ВПС1 следует, что когда два участника обьединяют доли, соответствующие белой точке, они не извлекают из этого никакой пользы.

С другой стороны, любые две строки матрицы M_B имеют вес r по ВПС2 и дают скалярное произведение λ по ВПС3. Данное свойство сохранится и после умножения M_B на матрицу перестановки. Из этого следует, что в объединении двух долей, соответствующих черной точке, $2r - \lambda$ единиц. В приведенном выше примере $r = 3$ и $\lambda = 1$, что дает $2r - \lambda = 5$ единиц в любом таком объединении. Можно сделать вывод, что у конструкции, созданной при помощи $\mathcal{M}_W$ и $\mathcal{M}_B$, верхнее визуальное пороговое значение β равно $(2r - \lambda)/m$.

Мы обосновали следующую общую конструкцию.

Конструкция 15.5
Пусть $n \times m$-матрица M_B обладает свойствами ВПС2 и ВПС3 для некоторых значений r и λ. Пусть матрица M_W имеет вид, заданный в ВПС1. Пусть теперь $\mathcal{M}_W$ и $\mathcal{M}_B$ получены из M_W и из M_B соответственно, умножением их справа на все возможные матрицы перестановок.
Тогда случайный выбор матрицы из $\mathcal{M}_W$ в случае разделения белой точки и случайный выбор матрицы из $\mathcal{M}_B$ в случае разделения черной приводит к визуальной $(n, 2)$-пороговой схеме с расширяющим множителем m и пороговыми значениями $\alpha = r/m$ и $\beta = (2r - \lambda)/m$.

Следствие 15.6
Возьмем произвольное n. Пусть u — некоторое число между 2 и $n - 1$. Пусть матрица M_B состоит из всех столбцов длины n и веса u. Тогда в M_B $m = \binom{n}{u}$ столбцов. Кроме того, вес каждой

строки из M_B равен $r = \binom{n-1}{u-1}$, и скалярное произведение любой пары строк из M_B равно $\lambda = \binom{n-2}{u-2}$.
Это определяет визуальную $(n, 2)$-пороговую схему с расширяющим множителем $m = \binom{n}{u}$ и пороговыми значениями $\alpha = u/m$ и $\beta = (2n - u + 1)/n(n-1)$.

Взяв в этом следствии $n = 4$ и $u = 2$, получим конструкцию из примера 15.4. Действительно, $m = \binom{n}{u} = \binom{4}{2} = 6$, $r = \binom{n-1}{u-1} = \binom{3}{1} = 3$ и $\lambda = \binom{n-2}{u-2} = \binom{2}{0} = 1$. Пороговые значения: $\alpha = 2/4 = 1/2$ и $\beta = 5/6$.

Недостатком семейства конструкций, описанных в приведенном выше следствии, является высокий расширяющий множитель m.

Читатель, знакомый с теорией блок-схем и t-схем[3], может по виду условий ВПС2 и ВПС3 догадаться, что эти понятия часто применяются при конструировании визуальных пороговых схем. Объясним одну такую конструкцию.

Пусть p — простое число. Напомним, что по определению А.9 целое число u, $1 \le u < p$, называется квадратичным вычетом (QR), если сравнение $x^2 \equiv u \pmod{p}$ имеет решение в $\mathbb{Z}_p$. Как по числу u определить, является ли оно квадратичным вычетом, объясняется в разд. А.4. В пакете "Mathematica" это можно сделать с помощью функции *JacobiSymbol*, которая возвращает 1 тогда и только тогда, когда u — квадратичный вычет.

Например, из

```
u = 12; m = 13; JacobiSymbol[u, m]
```

```
1
```

следует, что сравнение $x^2 \equiv 12 \pmod{13}$ имеет решение (это ± 5). Символ Якоби принято обозначать через $\left(\frac{u}{p}\right)$ или просто через $\chi(u)$, если не возникает путаницы со значением модуля p. По определению $\chi(u) = 0$ при $u = 0$, и $\chi(u) = -1$, когда $1 \le u < p$ и u не является QR.

Следствие 15.7
Пусть p — некоторое простое число, сравнимое с 3 по модулю 4. Определим $p \times p$-матрицу M_B как

$$(M_B)_{i,j} = \begin{cases} 1, & \text{если } j - i \text{ есть QR,} \\ 0 & \text{в противном случае.} \end{cases}$$

Тогда вес каждой строки из M_B равен $r = (p-1)/2$, а скалярное произведение любой пары строк равно $\lambda = (p-3)/4$.
Это определяет визуальную $(n, 2)$-пороговую схему с расширяющим множителем $m = n$ и пороговыми значениями $\alpha = (p-1)/2p$ и $\beta = (3p-1)/4p$.

[3] См., например, *[ЛидП96]. — *Прим. ред.*

Доказательство. Фиксируя номер строки i матрицы M_B, видим, что разности $j-i$, $0 \le j < p$, пробегают все элементы из $\mathbb{Z}_p$. Из теоремы А.20 следует, что вес каждой строки из M_B равен $(p-1)/2$.

Рассмотрим теперь матрицу $X = (\chi(j-i))_{0 \le i,j < p}$. Матрицу M_B можно получить из X заменой всех -1 на 0. Рассмотрим две строки матрицы X. Их номера — i_1 и i_2. Заметим, что

$$\chi(i_2 - i_1) \overset{\text{Теор. А.21}}{=} \chi(-1)\chi(i_1 - i_2) \overset{\text{След. А.24}}{=} -\chi(i_1 - i_2).$$

Т.е. матрица X — кососимметрическая и i_2-й элемент i_1-й строки противоположен i_1-му элементу i_2-й строки. Тогда после соответствующего переупорядочения координат рассматриваемые строки будут выглядеть так

$$\begin{array}{cccccc} \overbrace{0}^{1} & \overbrace{+1}^{1} & \overbrace{+1\ldots+1}^{a} & \overbrace{+1\ldots+1}^{b} & \overbrace{-1\ldots-1}^{c} & \overbrace{-1\ldots-1}^{d} \\ -1 & 0 & +1\ldots+1 & -1\ldots-1 & +1\ldots+1 & -1\ldots-1. \end{array}$$

Возможно, что строки тоже переставлены.

Скалярное произведение строк i_1 и i_2 матрицы M_B равно a (поскольку M_B получается из X заменой -1 на 0). Чтобы найти значения a, b, c, d, вычислим сперва

$$\sum_{j=0}^{p-1} \chi(j - i_1)\chi(j - i_2) = \sum_{j=0}^{p-1} \chi(j)\chi(j - (i_2 - i_1)) = -1. \qquad (15.4)$$

Первое равенство получается при подстановке $j - i_1 \mapsto j$, а второе следует из теоремы А.22, поскольку $i_1 \not\equiv i_2 \pmod p$.

Таким образом, имеем следующие соотношения:

$$\begin{array}{ll} 2 + a + b + c + d = p, & (\text{в } X \ p \text{ столбцов}), \\ a - b - c + d = -1, & (\text{из (15.4)}), \\ 1 + a + b = (p-1)/2, & (\text{применим теорему А.20 к первой строке}), \\ a + c = (p-1)/2 & (\text{применим теорему А.20 ко второй строке}). \end{array}$$

Эти уравнения имеют единственное решение: $a = b = d = (p-3)/4$ и $c = (p+1)/4$. Следовательно, скалярное произведение двух разных строк матрицы M_B равно $(p-3)/4$.

Теперь нужное нам следствие непосредственно вытекает из конструкции 15.5.

■

Пример 15.5

Возьмем $p = 11$. Матрицу M_B можно создать с помощью функций JacobiSymbol, If и Array пакета "Mathematica" следующим образом.

```
p = 11;
A[i_, j_] := If[JacobiSymbol[j - i, p] == 1, 1, 0];
MB = Array[A, {p, p}];
MatrixForm[MB]
```

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

Таким образом, имеем визуальную $(11, 2)$*-пороговую схему с расширяющим множителем* $m = 11$ *и пороговыми значениями* $\alpha = (p-1)/2p = 5/11$ *и* $\beta = (3p-1)/4p = 8/11$.

15.6 Задачи

Задача 15.1^M**.** Постройте $(5, 3)$-пороговую схему Шамира для секрета 15 над $GF(17)$. Покажите, как участники 1, 2 и 3 могут раскрыть секрет. Покажите, что при объединении участников 1 и 2 все элементы из $GF(17)$ будут одинаково правдоподобными кандидатами на роль секрета.

Задача 15.2^M**.** Рассмотрим $(5, 3)$-пороговую схему Шамира над $GF(23)$, в которой участники 1, 3, 4 и 6 объединяют свои доли $(1, 13)$, $(3, 19)$, $(4, 19)$ и $(6, 6)$ для раскрытия секрета S. Что это за секрет? Предположим, что участник 5 предъявляет в качестве своей доли пару $(5, 3)$. Почему один из этих пяти человек лжет? Пусть также и участники 1 и 8 предъявляют свои доли $(2, 4)$ и $(8, 12)$. Определите лжеца и настоящий секрет.

Задача 15.3^M**.** Постройте $(7, 4)$-пороговую схему над полем $GF(16) = GF(2)[a]/(a^4 + a + 1)$ (см. теорему B.15). Каковы доли участников, если секретом является вектор $S = (1, 0, 1, 1)$, соответствующий элементу поля a^{13}? Подробно покажите, как участники $2, 4, 5, 7$ раскрывают S.

Задача 15.4. Рассмотрим следующую схему над $\mathbb{Z}_3$:

УЧАСТНИК	ДОЛЯ
1	$a,\ b,\ c + s_2$
2	$a + s_1,\ b,\ c$
3	$b + s_1,\ c - s_2,\ d$
4	$b,\ d + s_2$

Дайте матричное описание этой схемы. Докажите, что это схема разделения секрета для структуры доступа $(U, \mathcal{P}, \mathcal{N})$, в которой $U = \{1,2,3,4\}$, $\mathcal{P}^- = \{\{1,2\},\{2,3\},\{3,1\},\{3,4\}\}$ и $\mathcal{N}^+ = \{\{1,4\},\{2,4\},\{3\}\}$. Какова информационная эффективность этой схемы? Совершенна ли она? Является ли она идеальной?

Задача 15.5. Постройте изображение множества возможных долей, возникающих при разделении черной точки в визуальной $(7,2)$-пороговой схеме, как в следствии 15.7. Каков расширяющий множитель этой схемы и каковы пороговые визуальные значения?

Приложение А

Элементарная теория чисел

А.1 Введение

Пусть $\mathbb{N}$ обозначает множество натуральных чисел, $\mathbb{Z}$ — множество целых чисел, а $\mathbb{R}$ — множество действительных чисел.

Целое число d *делит* целое число n, если $n = kd$ для некоторого $k \in \mathbb{Z}$. Это обозначается посредством $d|n$. Если такого целого k не существует, то d *не делит* n. Это обозначается через $d \not| n$.

Чтобы проверить, делит ли целое d целое число n, можно использовать функцию *IntegerQ* пакета "Mathematica". Например,

```
n = 16851; d = 123; IntegerQ[n / d]
```

```
True
```

Функция *Divisors* дает список всех натуральных делителей числа n. Например,

```
n = 16851; Divisors[n]
```

```
{1, 3, 41, 123, 137, 411, 5617, 16851}
```

Целое число $p, p > 1$, называется *простым*, если его единственными делителями являются 1 и p[1]. Введем естественную нумерацию на множестве простых чисел, полагая $p_1 = 2, p_2 = 3, p_3 = 5, \ldots$.

В этом контексте полезными функциями пакета "Mathematica" служат *Prime* и *PrimeQ*.

```
k = 35; Prime[k]
```

```
149
```

Этим порождается 35-е простое число.

```
n = 1234567; PrimeQ[n]
```

```
False
```

говорит, будет ли вход (в данном случае число 1234567) простым числом.

[1] Число $n > 1$, не являющееся простым, называется *составным*. — *Прим. ред.*

Теорема А.1 (*Эвклид*)
Существует бесконечно много простых чисел.

Доказательство. Предположим противное, и пусть $p_1, p_2, \ldots, p_k$ — множество всех простых чисел. Заметим, что целое число $\left(\prod_{i=1}^{k} p_i\right) + 1$ не делится ни на одно из простых чисел $p_1, p_2, \ldots, p_k$. Оно не может быть простым, так как не попадает в список $p_1, p_2, \ldots, p_k$. Отсюда следует, что n имеет нетривиальный делитель d. Но тогда d делится хотя бы на одно из простых $p_1, p_2, \ldots, p_k$, а потому делится и n. Противоречие.

■

Между двумя последовательными простыми числами может быть сколь угодно большая лакуна составных чисел. Например, $n-1$ элементов последовательности $n!+2, n!+3, \ldots, n!+n$ делятся, соответственно, на 2,3,...,n. Поэтому все они — составные.

Определение А.1
Функция $\pi(n)$ подсчитывает число простых чисел, не превосходящих n.

В пакете "Mathematica" эта функция обозначается PrimePi[n].

```
n = 100; PrimePi[n]
```

```
25
```

Следующая теорема, которую мы не будем доказывать[2], говорит об относительной частоте простых чисел в множестве $\mathbb{N}$.

Теорема А.2 (*закон распределения простых чисел*)

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

```
n = 1000000; PrimePi[n] / (n / Log[n]) // N
```

```
1.08449
```

Введем два важных понятия — *наибольшего общего делителя* и *наименьшего общего кратного* двух целых чисел.

Определение А.2
Наибольшим общим делителем двух целых чисел a и b, ни одно из которых не равно нулю[3], называется однозначно определенное положительное целое d, которое

[2] На русском языке имеется много книг, содержащих различные доказательства теоремы А.2. Укажем лишь три: *[Прах67], *[Чанд74] и *[Чер02]. — *Прим. ред.*

[3] Это условие излишне и в дальнейшем по существу игнорируется автором. — *Прим. ред.*

делит как a, так и b, $(A.1)$

и

если f делит a и b, то f делит d. $(A.2)$

Наибольший общий делитель чисел a и b обозначается через НОД(a, b) или просто (a, b).

Определение A.3
Наименьшим общим кратным двух целых чисел a и b называется однозначно определенное положительное целое m, которое

делится как на a, так и на b, $(A.3)$

и

если n делится на a и на b, то n кратно m. $(A.4)$

Наименьшее общее кратное чисел a и b обозначается через НОК$[a, b]$ или просто $[a, b]$.

Для доказательства существования НОД введем множество

$$U = \{x \cdot a + y \cdot b | x, y \in \mathbb{Z}, x \cdot a + y \cdot b > 0\}.$$

Пусть m — наименьший элемент в U. Покажем, что m удовлетворяет условиям (A.1) и (A.2). Очевидно, если f делит a и b, то f также делит m. Поэтому m удовлетворяет (A.2). Теперь запишем $a = qm + r$, $0 \leq r < m$, (вычитаем m из a или добавляем к a достаточное число раз, пока остаток r не будет лежать между 0 и $m - 1$). Если $r \neq 0$, то $r \in U$ (поскольку a и m лежат в U). Это противоречит предположению о минимальности m. Поэтому $r = 0$, и это означает, что m делит a. Аналогично, m делит b. Таким образом, m удовлетворяет также и (A.1).

Единственность НОД(a, b) вытекает из условий (A.1) и (A.2). В самом деле, если как d, так и d' удовлетворяют (A.1) и (A.2), то отсюда вытекает, что $d|d'$ и $d'|d$. Так как d и d' положительны, получаем, что $d = d'$.

Аналогичным путем могут быть доказаны существование и единственность НОК$[a, b]$.

Альтернативные определения НОД(a, b) и НОК$[a, b]$ таковы:

НОД(a, b) — наибольшее целое число, делящее как a, так и b.
НОК$[a, b]$ — наименьшее натуральное число, делящееся на a и b.

В пакете "Mathematica" имеются соответствующие функции *GCD* и *LCM*:

```
a = 12345; b = 67890; GCD[a, b]
```

```
15
```

```
a = 12345; b = 67890; LCM[a, b]
```

|| 55873470

Если у двух целых чисел НОД равен 1, то мы называем эти числа *взаимно простыми*. Следствием вышесказанного является следующая важная теорема.

Теорема А.3
Пусть a и b лежат в $\mathbb{N}$. Тогда существуют целые числа u и v, такие, что

$$\text{НОД}(a,b) = u \cdot a + v \cdot b.$$

В частности, если a и b взаимно просты, то существуют такие целые u и v, что

$$u \cdot a + v \cdot b = 1.$$

Следующая лемма выглядит слишком очевидной, чтобы нуждаться в доказательстве.

Лемма А.4
Пусть d делит произведение ab и НОД чисел d и a равен 1. Тогда d делит b.

Доказательство. Так как $\text{НОД}(d,a) = 1$, теорема А.3 влечет, что $xd + ya = 1$ для некоторых целых чисел x и y. Поэтому $xdb + yab = b$. Так как d делит ab, это влечет, что d также делит $xdb + yab$, равное b.

■

Следствие А.5
Пусть p — простое число, делящее $\prod_{i=1}^{k} a_i$, где $a_i \in \mathbb{Z}$, $1 \leq \leq i \leq k$. Тогда d делит по меньшей мере один из множителей a_i, $1 \leq i \leq k$.

Доказательство. Используйте лемму А.4 и индукцию по k.

■

Следующую теорему можно легко доказать индуктивным рассуждением.

Теорема А.6 (*основная теорема теории чисел*)
Любое натуральное число имеет единственное разложение вида

$$\prod_i p_i^{e_i}, \qquad e_i \in \mathbb{N}.$$

Пусть $a = \prod_i p_i^{e_i}$ и $b = \prod_i p_i^{f_i}$, где $e_i, f_i \in \mathbb{N}$. Тогда легко проверяется, что

$$\text{НОД}(a,b) = \prod_i p_i^{\min\{e_i,f_i\}}, \tag{A.5}$$

$$\text{НОК}[a,b] = \prod_i p_i^{\max\{e_i,f_i\}}, \tag{A.6}$$

$$\text{НОД}(a,b)\text{НОК}[a,b] = ab. \tag{A.7}$$

Выражение *FactorInteger*[n] из пакета "Mathematica" дает разложение целого числа n. Выходом служит список пар. Каждая пара содержит простой делитель числа n и его показатель.

```
FactorInteger[123456789]
```

```
|| {3, 2}, {3607, 1}, {3803, 1}
```

```
a = 21375; b = 89775;
FactorInteger[a]
FactorInteger[b]
FactorInteger[GCD[a, b]]
FactorInteger[LCM[a, b]]
GCD[a, b] * LCM[a, b] == a * b
```

```
|| {3, 2}, {5, 3}, {19, 1}
|| {3, 3}, {5, 2}, {7, 1}, {19, 1}
|| {3, 2}, {5, 2}, {19, 1}
|| {3, 3}, {5, 3}, {7, 1}, {19, 1}
|| True
```

A.2 Алгоритм Эвклида

Пусть a и b — два натуральных числа и $b \geq a$. Очевидно, любой делитель чисел a и b является делителем чисел a и $b-a$ и наоборот. Значит, НОД(a,b) = НОД$(a,b-a)$. Записав $b = q \cdot a + r$, по той же причине мы имеем НОД(a,b) = НОД(r,a). Если $r = 0$ (и $b = q \cdot a$), можно заключить, что НОД$(a,b) = a$, в противном случае мы продолжаем те же вычисления с a и r. Таким образом, можно записать $a = q' \cdot r + r'$, $0 \leq r' < r$, где НОД(a,b) = НОД(r',r), и т.д., пока один из аргументов не станет делить другой. Этот алгоритм является крайне быстрым способом вычисления НОД двух целых чисел и известен как *алгоритм Эвклида.*

Алгоритм A.7 (*простая версия алгоритма Эвклида*)

input натуральные числа a, b;
while $b > 0$ **do begin**
put r как остаток a при делении на b
(поэтому пишем $a = q \cdot b + r, 0 \leq r < b$);
put $a = b$;
put $b = r$;
end
output a.

Этот алгоритм реализуется в пакете "Mathematica" с помощью функций *While*, *Floor* и *Print*:

```
a = 1645; b = 861;
While[b != 0, r = a - Floor[a / b] * b;
  {a, b} = {b, r}; Print[{a, b}]]
```

```
{861, 784}
{784, 77}
{77, 14}
{14, 7}
{7, 0}
```

Если нужно также найти коэффициенты u и v, указанные в теореме А.3, то этот алгоритм может быть адаптирован, как описано ниже. Заметим, что при выбрасывании строк, включающих целые числа u_i и v_i, расширенный алгоритм сводится к простой версии алгоритма.

Алгоритм А.8 (*расширенная версия алгоритма Эвклида*)

input $b \geq a > 0$;
initialize $s_0 = b, s_1 = a,$
$u_0 = 0, u_1 = 1, v_0 = 1, v_1 = 0, n = 1;$
while $s_n > 0$ **do begin**
put $n = n + 1;$
write $s_{n-2} = q_n s_{n-1} + s_n, 0 \leq s_n < s_{n-1};$
put $u_n = q_n u_{n-1} + u_{n-2};$
put $v_n = q_n v_{n-1} + v_{n-2};$
end
put $u = (-1)^n u_{n-1}; v = (-1)^{n-1} v_{n-1}.$

$$\text{НОД}(a, b) = s_{n-1} = u \cdot a + v \cdot b. \qquad (A.8)$$

"Mathematica" содержит расширенную версию алгоритма Эвклида в качестве стандартной функции. Она называется *ExtendedGCD*.

```
a = 861; b = 1645; ExtendedGCD[a, b]
```

```
{7, {107, -56}}
```

Заметим, что в этом примере и в самом деле

$$7 = \text{НОД}(861, 1645) = 107 \times 861 - 56 \times 1645.$$

Доказательство корректности алгоритма А.8. Заметим сначала, что элементы $s_n, n \geq 1$, образуют строго убывающую последовательность неотрицательных целых чисел. Поэтому алгоритм заканчивается

самое большее после b итераций. Позже в этом разделе мы проанализируем, насколько быстр алгоритм Эвклида на самом деле.

Из рекуррентного соотношения $s_k = s_{k-2} - q_k s_{k-1}$ алгоритма следует, что

$$\text{НОД}(a,b) = \text{НОД}(s_0,s_1) = \text{НОД}(s_1,s_2) = \ldots = \text{НОД}(s_{n-1},s_n) =$$

$$= \text{НОД}(s_{n-1},0) = s_{n-1}.$$

Это доказывает первое равенство в А.8. Докажем теперь, что для всех $k, 0 \le k \le n$,

$$(-1)^{k-1}u_k a + (-1)^k v_k b = s_k.$$

Заметим, что подстановка $k = n-1$ в это соотношение доказывает второе равенство в А.8.

Для $k = 0$ и $k = 1$ соотношение выше выполняется благодаря нашему выбору начальных значений для u_0, u_1, v_0 и v_1. Теперь мы применим индукцию. Из рекуррентных соотношений алгоритма и из индуктивной гипотезы следует, что

$$s_k = s_{k-2} - q_k s_{k-1} =$$

$$= \{(-1)^{k-3}u_{k-2}a + (-1)^{k-2}v_{k-2}b\} - q_k\{(-1)^{k-2}u_{k-1}a + (-1)^{k-1}v_{k-1}b\} =$$

$$= (-1)^{k-1}(u_{k-2}+q_k u_{k-1})a + (-1)^k(v_{k-2}+q_k v_{k-1})b = (-1)^{k-1}u_k a + (-1)^k v_k b.$$

Конечно, нет необходимости записывать ранее вычисленные в программе значения s_k, u_k и v_k. Достаточно только каждых последних двух вместе с q_k. Причиной введения их в алгоритм было облегчение чтения доказательства.

■

Приведем этот алгоритм в пакете "Mathematica", используя функции *While*, *Floor* и *Print*.

```
b = 1645; a = 861;
n = 1;
so = b; sn = a;
uo = 0; un = 1;
vo = 1; vn = 0;
While[sn != 0, Print[(-1)^(n-1), "x", un, "x",
    a, " + ", (-1)^n, "x", vn, "x", b, "=", sn];
    q = Floor[so / sn]; n = n + 1;
    {so, sn, uo, un, vo, vn} =
    {sn, so - q * sn, un, q * un + uo, vn, q * vn + vo}]
```

$1 \times 1 \times 861 + -1 \times 0 \times 1645 = 861$
$-1 \times 1 \times 861 + 1 \times 1 \times 1645 = 784$
$1 \times 2 \times 861 + -1 \times 1 \times 1645 = 77$

$-1 \times 21 \times 861 + 1 \times 11 \times 1645 = 14$
$1 \times 107 \times 861 + -1 \times 56 \times 1645 = 7$

Мы хотим завершить этот раздел, сказав несколько слов о сложности алгоритма Эвклида. Должно быть ясно, что этот алгоритм наиболее медленен, когда на каждом шаге q_k принимает значение 1 (если это возможно). Это случай, в котором $s_{k-2} = s_{k-1} + s_k$ для всех $k, 2 \leq k \leq n-1$, и $s_{n-2} = 2s_{n-1}, s_n = 0$. Другими словами, наименьшие значения b и a, требующие $n-1$ шагов, даются равенствами $b = F_n$ и $a = F_{n-1}$, где $F_i, i \geq 0$, — известная последовательность *чисел Фибоначчи*, определяемая равенствами $F_0 = 0, F_1 = 1, F_{i+2} = F_{i+1} + F_i$ для $i \geq 0$.

Позволяя пакету "Mathematica" оперировать одновременно с двумя последовательными числами Фибоначчи (для этого используется функция *Nest*) мы получаем следующий метод вычисления этих чисел (в примере вычисляются F_{100} и F_{101}):

```
f[{u_, v_}] := {v, u + v};
n = 100; Nest[f, {0, 1}, n]
```

```
|| {354224848179261915075, 573147844013817084101}
```

Это можно сделать также прямо, используя функцию *Fibonacci*.

```
Fibonacci[100]
```

```
|| 354224848179261915075
```

Читатель может проверить вышесказанное следующим образом.

```
GCDiterations[n_Integer?Positive, m_Integer?Positive] :=
  Block[{a = n, b = m, r, t = 0},
  While[b > 0, r = Mod[a, b]; {a, b, t} = {b, r, t + 1}];
    t]
```

```
n = 100;
GCDiterations[Fibonacci[n], Fibonacci[n - 1]]
```

```
|| 98
```

```
Table[GCDiterations[Fibonacci[n], Fibonacci[n - 1]],
  {n, 2, 100}]
```

```
|| {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
   17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
   31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
   45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58,
   59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72,
   73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,
   87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98}
```

Заметим, что алгоритм GCDiterations, использованный выше, не влияет на значения a и b (в противовес нашей реализации простой версии алгоритма Эвклида). Он также использует функцию *Mod* из пакета "Mathematica", которую мы обсудим в следующем разделе.

Подставляя $F_n = cf^n$ в определяющее рекуррентное соотношение для чисел Фибоначчи, $F_{i+2} = F_{i+1} + F_i$, получаем квадратное уравнение $f^2 = f + 1$, корни которого суть $(1 \pm \sqrt{5})/2$. Мы укажем без доказательства следующую верхнюю границу сложности алгоритма Эвклида.

Теорема A.9
Пусть a и b — натуральные числа, $b \geq a$, $b \neq 1$, и пусть $f = (1 + \sqrt{5})/2$. Тогда число итераций, требуемых алгоритмом Эвклида при вычислении НОД(a, b), не превосходит $\log_f b$.

```
a = Fibonacci[100]; b = Fibonacci[99];
GCDiterations[a, b]
Ceiling[Log[(1 + Sqrt[5]) / 2, b]]
```

```
98
98
```

A.3 Сравнения, теоремы Эйлера, Ферма и китайская теорема об остатках

A.3.1 Сравнения

Определение A.4
Два целых числа a и b называются *сравнимыми* друг с другом *по модулю* m, если их разность $b - a$ делится на m. Это обозначается формулой

$$a \equiv b \pmod{m}.$$

Функция *Mod*$[a, m]$ из пакета "Mathematica" дает единственное целое $r, 0 \leq r < m$, такое, что $a \equiv r \pmod{m}$.

```
a = 12345; m = 13; Mod[a, m]
```

```
8
```

Следующий пример показывает, как легко проверить сравнимость двух целых чисел по модулю m:

```
m = 13; a = 12345; b = 103579; Mod[a - b, m] == 0
```

```
True
```

Определение А.5
Множество из m целых чисел $a_1, a_2, \ldots, a_m$ называется *полной системой вычетов*, если каждое целое число сравнимо (в точности) с одним из элементов $a_i, 1 \leq i \leq m$, по модулю m.

Чаще всего используемые полные системы вычетов по модулю m суть $\{0, 1, \ldots, m-1\}$ и $\{1, 2, \ldots, m\}$. Эти системы можно породить функциями *Range* и *Table* из пакета "Mathematica".

```
m = 10;
Table[i, {i, 0, m - 1}]
Range[m]
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Очевидно, что m целых чисел $a_i, 1 \leq i \leq m$, образуют полную систему вычетов по модулю m тогда и только тогда, когда для любой пары $i, j, 1 \leq i, j \leq m$, выполняется

$$a_i \equiv a_j \pmod{m} \Longrightarrow i = j. \qquad (A.9)$$

Отношение сравнения является отношением эквивалентности (см. определение В.5) на $\mathbb{Z}$. Полная система вычетов есть просто множество представителей m классов эквивалентности.

Лемма А.10
Пусть $ka \equiv kb \pmod{m}$ и $\text{НОД}(k, m) = d$. Тогда

$$a \equiv b \pmod{m/d}.$$

Доказательство. Запишем $k = k'd$ и $m = m'd$, где $\text{НОД}(k', m') = 1$. Из того, что $ka - kb = xm$ для некоторого $x \in \mathbb{Z}$, следует, что $k'(a-b) = xm'$. Так как $\text{НОД}(m', k') = 1$, лемма А.4 влечет, что $m' | (a-b)$, т.е. $a \equiv b \pmod{m'}$.

■

Лемма А.11
Пусть $a_1, a_2, \ldots, a_m$ — полная система вычетов по модулю m и пусть $\text{НОД}(k, m) = 1$. Тогда $ka_1, ka_2, \ldots, ka_m$ — тоже полная система вычетов по модулю m.

Доказательство. Используем критерий (А.9). По лемме А.10 $ka_i \equiv ka_j \pmod{m}$ влечет, что $a_i \equiv a_j \pmod{m}$. Это, в свою очередь, влечет, что $i = j$.

■

А.3.2 Теоремы Эйлера и Ферма

Мы часто будем интересоваться только теми представителями классов вычетов по модулю m, чьи элементы взаимно просты с m. Число этих классов дается следующей функцией.

Определение А.6
Функция Эйлера определяется равенством

$$\varphi(m) = |\{i | 0 \leq i < m, \text{НОД}(i, m) = 1\}|.$$

Иными словами, $\varphi(m)$ — это число целых чисел между 0 и $m-1$, которые взаимно просты с m.

В пакете "Mathematica" этой функции отвечает *EulerPhi*. Например,

```
m = 15; EulerPhi[m]
```

```
8
```

Соответствующие восемь элементов — 1, 2, 4, 7, 8, 11, 13 и 14. Позже в этом разделе мы увидим, как можно эффективно вычислять $\varphi(m)$.

Теорема А.12
Для всех натуральных чисел m

$$\sum_{d|m} \varphi(d) = m.$$

Это довольно легко увидеть на примере, когда из m целых чисел между 1 и m берутся только члены $\varphi(d)$, $d|m$. Когда $m = 15$, делителями m служат 1, 3, 5 и 15. Восемь элементов 1, 2, 4, 7, 8, 11, 13, 14 имеют с числом 15 НОД 1 (напомним, $\varphi(15) = 8$), четыре ($= \varphi(5)$) элемента 3, 6, 9, 12 имеют с 15 НОД $= 3$, два ($= \varphi(3)$) элемента 5 и 10 имеют НОД $= 5$ и один ($= \varphi(1)$) элемент 0 имеет НОД $= 15$.

Доказательство теоремы А.12. Пусть d делит m. При записи $r = id$ непосредственно видно, что число элементов $r, 0 \leq r < m$, с $\text{НОД}(r, m) = d$ равно числу целых чисел i, таких, что $0 \leq i < \frac{m}{d}$ и $\text{НОД}(i, \frac{m}{d}) = 1$; поэтому таким числом будет $\varphi(\frac{m}{d})$.

С другой стороны, $\text{НОД}(r, m)$ делит m для каждого целого $r, 0 \leq r < m$. Отсюда следует, что $\sum_{d|m} \varphi(\frac{m}{d}) = m$. Это эквивалентно доказываемому утверждению.

■

Следующее нестандартное предложение пакета "Mathematica" вычисляет сумму значений $f(d)$ по всем делителям d целого числа m:

```
DivisorSum[f_, m_] := Plus @@ (f / @ Divisors[m])
```

Эту функцию можно использовать для проверки теоремы А.12:

```
m = 15; DivisorSum[EulerPhi, m]
```

```
15
```

Определение А.7
Множество из $\varphi(m)$ целых чисел $r_1, r_2, \ldots, r_{\varphi(m)}$ называется *приведенной системой вычетов* по модулю m, если каждое целое число j с НОД$(j, m) = 1$ сравнимо ровно с одним элементом $r_i, 1 \leq i \leq \varphi(m)$.

Приведенная система вычетов может легко порождаться с помощью следующих заново определенных функций.

```
CoPrimeQ[n_Integer, m_Integer] := GCD[n, m] == 1
```

```
CoPrimeQ[35, 91]
CoPrimeQ[36, 91]
```

```
False
```

```
True
```

```
CoPrimes[n_Integer?Positive] :=
    Select[Range[n], CoPrimeQ[n, #] &]
```

```
CoPrimes[15]
```

```
{1, 2, 4, 7, 8, 11, 13, 14}
```

Имеется следующая лемма, аналогичная лемме А.11.

Лемма А.13
Пусть $r_1, r_2, \ldots, r_\varphi$ — приведенная система вычетов по модулю m, и НОД$(a, m) = 1$. Тогда $ar_1, ar_2, \ldots, ar_\varphi$ — тоже приведенная система вычетов по модулю m.

С помощью этой леммы легко доказать, что классы в приведенной системе вычетов образуют группу (см. подраздел В.1.1).

Теорема А.14 (*Эйлер*)
Пусть a и m — взаимно простые целые числа. Тогда

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

Эту теорему довольно легко проверить в конкретных случаях.

```
m = 12345; a = 11111; GCD[m, a]
EulerPhi[m]
Mod[a^EulerPhi[m], m]
```

```
|| 1

|| 6576

|| 1
```

Возведение в степень по модулю некоторого целого числа в пакете "Mathematica" может выполняться много быстрее с помощью функции *PowerMod*, которая приводит все промежуточные результаты при вычислении a^b по модулю n.

```
m = 123456789; a = 1111111111; GCD[m, a]
PowerMod[a, EulerPhi[m], m]
```

```
|| 1

|| 1
```

Доказательство теоремы А.14. Пусть $r_1, r_2, \ldots, r_\varphi$ — приведенная система вычетов по модулю m. По лемме А.11

$$\prod_{i=1}^{\varphi(m)} r_i \equiv \prod_{i=1}^{\varphi(m)} (ar_i) \equiv a^{\varphi(m)} \prod_{i=1}^{\varphi(m)} r_i \pmod{m}.$$

Так как каждый делитель r_i взаимно прост с m, можно разделить обе части равенства на $\prod_{i=1}^{\varphi(m)} r_i$ в силу леммы А.10. Результатом будет $1 \equiv \equiv a^{\varphi(m)} \pmod{m}$.

■

Пусть p — простое число. Так как любое целое $i, 1 \leq i < p$, взаимно просто с p, это показывает, что $\varphi(p) = p - 1$. Теорема Эйлера влечет следующую теорему для всех значений a.

Теорема А.15 (*малая теорема Ферма*)
Пусть p — простое число, a — произвольное целое. Тогда

$$a^p \equiv a \pmod{p}.$$

В конкретных случаях это легко проверить в пакете "Mathematica" с помощью функции *PowerMod*.

```
p = 98947; a = 12345; PrimeQ[p]
PowerMod[a, p, p] == a
```

```
|| True

|| True
```

Как мы заметили, $\varphi(p) = p - 1$ для простого p. Поскольку в точности одно из любых p последовательных целых чисел делится на p, мы получаем следующий более сильный результат:

$$\varphi(p^e) = p^e - (p^e/p) = p^{e-1}(p-1) = p^e(1 - \frac{1}{p}). \qquad (A.10)$$

Определение А.8
Функция $f : \mathbb{N} \to \mathbb{N}$ называется *мультипликативной*, если для любой пары натуральных чисел m и n

$$\text{НОД}(m,n) = 1 \Longrightarrow f(m \cdot n) = f(m)f(n).$$

Лемма А.16
Функция Эйлера φ мультипликативна.

Доказательство. Пусть m и n взаимно просты и пусть $a_1, a_2, \ldots, a_{\varphi(m)}$ и $b_1, b_2, \ldots, b_{\varphi(n)}$ — приведенные системы вычетов по модулю m и n соответственно. Достаточно показать, что $\varphi(m)\varphi(n)$ целых чисел $n \cdot a_i + m \cdot b_j$, $1 \le i \le \varphi(m)$ и $1 \le j \le \varphi(n)$, образуют приведенную систему вычетов по модулю mn. Легко проверить, что целые числа $n \cdot a_i + m \cdot b_j$, $1 \le i \le \varphi(m)$ и $1 \le j \le \varphi(n)$, все различны по модулю mn и что они взаимно просты с mn. (Используйте лемму А.15 и формулу (А.9)).

Остается проверить, что любое целое число k с $\text{НОД}(k, mn) = 1$ сравнимо с $n \cdot a_i + m \cdot b_j$ по модулю mn для некоторых i, $1 \le i \le \varphi(m)$, и j, $1 \le j \le \varphi(n)$. Из леммы А.13 мы знаем, что существуют целые числа i и j, $1 \le i \le \varphi(m)$, $1 \le j \le \varphi(n)$, такие, что

$$k \equiv n \cdot a_i \pmod{m}, \quad k \equiv m \cdot b_j \pmod{n}.$$

Это влечет, что как m, так и n делят $k - n \cdot a_i - m \cdot b_j$. Поскольку $\text{НОД}(m,n) = 1$, из условий (А.4) и (А.7) следует, что $m \cdot n$ также делит $k - n \cdot a_i - m \cdot b_j$.

■

Теорема А.17

$$\varphi(m) = m \prod_{p \text{ простое},\ p|m} (1 - \frac{1}{p}).$$

Доказательство. Скомбинируйте формулу (А.10) и лемму А.16.

■

В разделе А.5 мы увидим, как теорема А.17 доказывается прямым вычислительным рассуждением.

С помощью функций *Length* и *EulerPhi* из "Mathematica" и определенной выше функции *CoPrimes* (с использованием *CoPrimeQ*) можно проверить теорему А.17 следующим образом:

```
m = 15;
Length[CoPrimes[m]]
EulerPhi[m]
```

```
8
```

```
8
```

A.3.3 Решение линейных сравнений

Простейшее сравнение, которое может быть решено, — это *линейное сравнение*

$$ax \equiv b \pmod{m}. \tag{A.11}$$

Теорема A.18
Линейное сравнение $ax \equiv b \pmod{m}$ имеет решение x тогда и только тогда, когда НОД(a, m) делит b.
В этом случае число различных решений по модулю m равно НОД(a, m).

Доказательство. То, что условие НОД$(a, m)|b$ необходимо, чтобы (A.11) имело решение x, тривиально. Докажем теперь, что оно также и достаточно.

Пусть $d =$ НОД(a, m), тогда запишем $a = a'd$, $m = m'd$ и $b = b'd$, где НОД$(a', m') = 1$. По лемме A.11 сравнение $a'x \equiv b' \pmod{v'}$ имеет единственное решение x' по модулю m'. Очевидно, решение x сравнения $ax \equiv b \pmod{m}$ удовлетворяет сравнению $x \equiv x' \pmod{m'}$. Таким образом, каждое решение x по модулю m может быть записано в виде $x' + im', 0 \leq i < d$. Запишем $a'x' = b' + um', u \in \mathbb{Z}$. Тогда для каждого $i, 0 \leq i < d$,

$$a(x' + im') = da'x' + ida'm' = db' + udm' + ia'm = b + (u + ia')m.$$

Следовательно, числа $x' + im', 0 \leq i < d$, представляют все решения по модулю m сравнения $ax \equiv b \pmod{m}$.

■

Решение сравнения $ax \equiv b \pmod{m}$ легко найти с помощью расширенной версии алгоритма Эвклида. В самом деле, из $ua + vb = 1$ (см. теорему A.3) следует, что $ua \equiv 1 \pmod{m}$. Таким образом, решение x дается как $bu \pmod{m}$. Если НОД$(a, m) = 1$, то обычно пишется a^{-1} для единственного u, удовлетворяющего сравнению $ua \equiv 1 \pmod{m}$.

Пример A.1 (метод 1)

Чтобы решить сравнение $14x \equiv 26 \pmod{34}$, *заметим, что* НОД$(14, 34) = 2$, *которое действительно делит* 26.

Сначала мы решаем сравнение $7x' \equiv 13 \pmod{17}$. *С помощью расширенной версии алгоритма Эвклида находим* $5 \cdot 7 + (-2)17 =$ НОД$(7,$

17) = 1. *Поэтому* $7 \cdot 5 \equiv 1 \pmod{17}$ *и* x' *может быть вычислен из сравнения* $x' \equiv 7^{-1} \cdot 13 \equiv 5 \cdot 13 \equiv 14 \pmod{17}$.

По теореме выше $14x \equiv 26 \pmod{34}$ *имеет решения* 14 *и* $14 + 17 = 31$ *по модулю* 34.

```
ExtendedGCD[7, 17]
Mod[5 * 13, 17]
```

```
{1, {5, -2}}
```

```
14
```

Пример А.2 (метод 2)

Чтобы решить сравнение $123456789x \equiv 135798642 \pmod{179424673}$, *мы сначала проверяем, будет ли* НОД(123456789, 179424673) *делить* 135798642. *Затем мы вычисляем* $123456789^{-1} \pmod{179424673}$ *и вычисляем* $123456789^{-1} \cdot 135798642$, *что дает* 21562478 *в качестве решения.*

Для вычисления $123456789^{-1} \pmod{179424673}$ *вместо алгоритма Эвклида можно также использовать теорему Эйлера. В самом деле,* $a^{\varphi(m)} \equiv 1 \pmod{m}$ *влечет, что* $a\,a^{\varphi(m)-1} \equiv 1 \pmod{m}$ *и поэтому* $a^{-1} \equiv a^{\varphi(m)-1} \pmod{m}$.

```
GCD[123456789, 179424673]
PowerMod[123456789, EulerPhi[179424673] - 1, 179424673]
```

```
1
```

```
172609538
```

Таким образом, число 172609538 — *мультипликативное обратное к* 123456789 *по модулю* 179424673. *Решение* x *сравнения* $123456789x \equiv 135798642 \pmod{179424673}$ *получается следующим образом:*

```
Mod[123456789 * 172609538, 179424673]
```

```
21562478
```

Это можно проверить:

```
Mod[123456789 * 21562478, 179424673]
```

```
135798642
```

Функция *PowerMod* из пакета "Mathematica" очень эффективно вычисляет мультипликативное обратное к данному числу:

```
PoweerMod[123456789, -1, 179424673]
```

```
172609538
```

Функция *Solve* пакета "Mathematica" дает все решения сравнения $ax \equiv b \pmod{m}$, если они существуют.

```
Clear[x];
Solve[{12 x == 8, Modulus == 16}, x]
```

```
{{Modulus -> 16, x -> 2}, {Modulus -> 16, x -> 6},
{Modulus -> 16, x -> 10}, {Modulus -> 16, x -> 14}}
```

Чтобы получить только решения, можно выполнить

```
x /. Solve[{12 x == 8, Modulus == 16}, x]
```

```
{2, 6, 10, 14}
```

Мы приглашаем читателя испытать

```
x / . Solve[{13 x == 1, Modulus == 16}, x]
```

```
Solve[{12 x == 7, Modulus == 17}, x]
```

А.3.4 Китайская теорема об остатках

Здесь мы обсудим случай, когда x одновременно удовлетворяет нескольким линейным сравнениям, скажем, $a_i x \equiv b_i \pmod{m_i}$, с $\text{НОД}(a_i, m_i) | b_i$ для $1 \leq i \leq k$. Деля i-е сравнение на $d_i = \text{НОД}(a_i, m_i)$, $1 \leq i \leq k$, получаем, как и раньше, сравнения $a_i' x' \equiv b_i' \pmod{m_i'}$ с $\text{НОД}(a_i', m_i') = 1$. В силу доказательства теоремы А.18 решение этого сравнения эквивалентно решению одного из d сравнений $a_i' x \equiv b_i' + j m_i' \pmod{m_i}, 0 \leq j < d$. С этой точки зрения мы можем ограничить наше внимание тем случаем, когда $\text{НОД}(a_i, m_i) = 1$ для всех $i, 1 \leq i \leq k$.

Теорема А.19 (*китайская теорема об остатках*)
Пусть $m_i, 1 \leq i \leq k$, — попарно взаимно простые натуральные числа. Далее, пусть $a_i, 1 \leq i \leq k$, — целые числа с $\text{НОД}(a_i, m_i) = 1$. Тогда система k сравнений

$$a_i x \equiv b_i \pmod{m_i}, \qquad 1 \leq i \leq k, \tag{A.12}$$

имеет *единственное решение* по модулю $\prod_{i=1}^{k} m_i$ для всех возможных k-ок целых чисел $b_1, b_2, \ldots, b_k$.

Доказательство. Допустим, что x' и x'' — два решения системы. Тогда $a_i(x' - x'') \equiv 0 \pmod{m_i}, 1 \leq i \leq k$. По лемме А.4 m_i делит $x' - x''$ для всех $i, 1 \leq i \leq k$. Отсюда следует, что $x' \equiv x'' \pmod{\prod_{i=1}^{k} m_i}$. Следовательно, k сравнений имеют единое решение, и оно единственно по модулю $\prod_{i=1}^{k} m_i$.

С другой стороны, так как имеется столько же значений x по модулю $\prod_{i=1}^{k} m_i$, сколько и возможных приведенных k-ок $b_1, b_2, \ldots, b_k$, между ними должно быть взаимно однозначное соответствие.

■

Наше доказательство не дает эффективного алгоритма для нахождения решения системы (А.12). Мы сейчас объясним, как это может быть сделано.

Пусть $1 \leq i \leq k$ и u_i — единственное решение по модулю $\prod_{i=1}^{k} m_i$ системы

$$a_i u_i \equiv 1 \pmod{m_i}, \tag{A.13}$$

$$a_j u_i \equiv 0 \pmod{m_i}, \qquad 1 \leq j \leq k, \qquad j \neq i. \tag{A.14}$$

С помощью алгоритма Эвклида u_i легко определяются. В самом деле, из (А.14) следует, что u_i — кратное числа $m^{(i)}$, определенного как $\prod_{j, j \neq i} m_j$, скажем, $u_i = r m^{(i)}$ для некоторого $r, 0 \leq r < m_i$. Значение r получается из (А.13). Действительно, r является решением сравнения $a_i r m^{(i)} \equiv 1 \pmod{m_i}$. Следовательно,

$$u_i = \{(a_i m^{(i)})^{-1} \bmod m_i\} m^{(i)}.$$

Числа $u_i, 1 \leq i \leq k$, можно записать, используя самое большее $k \log_2 m$ битов пространства памяти.

Решение системы (А.12) дается теперь формулой

$$x = u_1 b_1 + u_2 b_2 + \ldots + u_k b_k.$$

Пример А.3

Чтобы решить систему

$$3x \equiv 7 \pmod{11}, \quad 2x \equiv 9 \pmod{13}, \quad 12x \equiv 5 \pmod{17},$$

перепишем эти сравнения в виде

$$x \equiv 3^{-1} \cdot 7 \pmod{11}, \quad x \equiv 2^{-1} \cdot 9 \pmod{13}, \quad x \equiv 12^{-1} \cdot 5 \pmod{17},$$

которые приводятся к

$$x \equiv 4 \cdot 7 \pmod{11}, \quad x \equiv 7 \cdot 9 \pmod{13}, \quad x \equiv 10 \cdot 5 \pmod{17},$$

т.е.

$$x \equiv 6 \pmod{11}, \quad x \equiv 11 \pmod{13}, \quad x \equiv 16 \pmod{17}.$$

Теперь мы найдем решения сравнений

$$\begin{array}{lll} u_1 \equiv 1 \pmod{11}, & u_1 \equiv 0 \pmod{13}, & u_1 \equiv 0 \pmod{17} \\ u_2 \equiv 0 \pmod{11}, & u_2 \equiv 1 \pmod{13}, & u_2 \equiv 0 \pmod{17} \\ u_3 \equiv 0 \pmod{11}, & u_3 \equiv 0 \pmod{13}, & u_3 \equiv 1 \pmod{17} \end{array}$$

Записывая $u_1 = l_1 \cdot 13 \cdot 17$, $u_2 = l_2 \cdot 11 \cdot 17$, $u_3 = l_3 \cdot 11 \cdot 13$, *мы с помощью теоремы A.18 (или с помощью функции Solve) находим, что* $l_1 \equiv 1 \pmod{11}$, $l_2 \equiv 8 \pmod{13}$, $l_3 \equiv 5 \pmod{17}$ *и поэтому* $u_1 \equiv 221 \pmod{11 \cdot 13 \cdot 17}$, $u_2 \equiv 1496 \pmod{11 \cdot 13 \cdot 17}$, $u_3 \equiv 715 \pmod{11 \cdot 13 \cdot 17}$.

Мы заключаем, что $x \equiv 6 \cdot 221 + 11 \cdot 1496 + 16 \cdot 715 \equiv 50 \pmod{11 \cdot 13 \cdot 17}$.

Чтобы решить сравнения $x \equiv b_i \pmod{m_i}, 1 \leq i \leq k$, где все m_i взаимно просты, с помощью китайской теоремы об остатках в системе "Mathematica", нужно сначала загрузить пакет *NumberTheory'NumberTheoryFunctions'*.

```
<< NumberTheory'NumberTheoryFunctions'
```

Теперь система сравнений может быть решена с помощью функции *ChineseRemainderTheorem*, доступной в указанном пакете. Мы продемонстрируем это, определяя u_1, u_2 и u_3 из приведенного выше примера.

```
ChineseRemainderTheorem[{1, 0, 0}, {11, 13, 17}]
ChineseRemainderTheorem[{0, 1, 0}, {11, 13, 17}]
ChineseRemainderTheorem[{0, 0, 1}, {11, 13, 17}]
```

```
|| 221
|| 1496
|| 715
```

При рассмотрении системы сравнений $a_i x \equiv b_i \pmod{m_i}, 1 \leq i \leq k$, где все m_i взаимно просты и НОД$(a_i, m_i) = 1$ для $1 \leq i \leq k$ в "Mathematica" легко привести эту систему к эквивалентной системе $x \equiv a_i^{-1} b_i \pmod{m_i}, 1 \leq i \leq k$, которая может быть решена с помощью китайской теоремы об остатках. Для этого сведéния используются функции *PowerMod* и *Mod*. Они действуют одинаково как на числах, так и на векторах (покоординатно).

Продемонстрируем это с параметрами из предыдущего примера.

```
a = {3, 2, 12}; b = {7, 9, 5}; m = {11, 13, 17}
b = Mod[b * PowerMod[a, -1, m], m]
ChineseRemainderTheorem[b, m]
```

```
|| {6, 11, 16}
|| 50
```

A.4 Квадратичные вычеты

Пусть p — нечетное простое число. Квадратичное сравнение $ax^2 + bx + c \equiv 0 \pmod{p}$, $a \not\equiv 0 \pmod{p}$ можно упростить, разделив его на a, а

затем применить подстановку $x \mapsto x - b/(2a)$. Таким путем $ax^2 + bx + c \equiv \equiv 0 \pmod{p}$ приводится к квадратичному сравнению типа

$$x^2 \equiv u \pmod{p}. \tag{A.15}$$

Определение А.9
Пусть p — нечетное простое число, а u — целое число, не делящееся на p. Число u называется *квадратичным вычетом* (QR) по модулю p, если сравнение (A.15) имеет целочисленное решение, и называется *квадратичным невычетом* (NQR), если (A.15) не имеет целочисленного решения.

Определение А.10
Пусть p — нечетное простое число, а u — целое. *Символ Лежандра* $\left(\frac{u}{p}\right)$ определяется равенством

$$\left(\frac{u}{p}\right) = \begin{cases} +1, & \text{если } u - \text{ квадратичный вычет по модулю } p, \\ -1, & \text{если } u - \text{ квадратичный невычет по модулю } p, \\ 0, & \text{если } p \text{ делит } u. \end{cases}$$

Если нет сомнений в реальном выборе простого числа p, то вместо $\left(\frac{u}{p}\right)$ часто пишут $\chi(u)$.

Символ Лежандра — это частный случай следующей функции.

Определение А.11
Пусть $m = \prod_i p_i^{e_i}$ — нечетное целое число, а u — целое число с НОД$(u, m) = 1$. Тогда *символ Якоби* $\left(\frac{u}{m}\right)$ определяется равенством

$$\left(\frac{u}{m}\right) = \prod_i \left(\frac{u}{p_i}\right)^{e_i},$$

где $\left(\frac{u}{p_i}\right)$ обозначает символ Лежандра.

Символ Якоби (а, значит, и символ Лежандра) может вычисляться с помощью стандартной функции *JacobiSymbol* из пакета "Mathematica". Например, мы можем проверить, будет ли 12 квадратичным вычетом по модулю 13 (на самом деле, $5^2 \equiv 12 \pmod{13}$) посредством вычисления *JacobiSymbol*[12,13], что дает значение 1:

```
u = 12; m = 13; JacobiSymbol[u, m]
```

```
1
```

Мы хотим вывести некоторые свойства символа Лежандра.

Пусть $a^2 \equiv u \pmod{p}$. Тогда также $(p-a)^2 \equiv u \pmod{p}$. Многочлен $x^2 - u$ имеет самое большее два нуля в GF(p) (см. теорему B.15),

поэтому по модулю p не может быть более двух различных решений сравнения $x^2 \equiv u \pmod{p}$. Отсюда следует, что квадратичные вычеты по модулю p — это целые числа

$$i^2 \bmod p, \quad 1 \le i \le \frac{p-1}{2},$$

или, альтернативно, целые числа $(p-i)^2, 1 \le i \le \frac{p-1}{2}$. Мы заключаем, что имеется в точности $\frac{p-1}{2}$ QR и $\frac{p-1}{2}$ NQR. Это доказывает первую из следующих двух теорем.

Теорема A.20
Пусть p — нечетное простое число. Тогда ровно $\frac{p-1}{2}$ среди чисел $1, 2, \ldots, p-1$ — квадратичные вычеты и $\frac{p-1}{2}$ — квадратичные невычеты. Имеем формулу

$$\sum_{u=0}^{p-1} \chi(u) = 0.$$

Читатель может проверить эту теорему на конкретных примерах с помощью двух следующих функций пакета "Mathematica".

```
p = 17; ∑_{i=0}^{p-1} JacobiSymbol[i, p]
```

```
0
```

```
ListQuadRes[p_] :=
  Select[Range[p], JacobiSymbol[#1, p] == 1 &]
```

```
p = 17;
ListQuadRes[p]
```

```
{1, 2, 4, 8, 9, 13, 15, 16}
```

Теорема A.21
Пусть p — нечетное простое. Тогда для всех целых чисел u и v

$$\chi(u \cdot v) = \chi(u) \cdot \chi(v).$$

Доказательство. Эта теорема окажется тривиальным следствием доказываемой позже теоремы A.23. Мы представим здесь более элементарное доказательство.

Если p делит u или v, то утверждение тривиально, потому что обе части равенства равны нулю. В случае, когда p не делит ни u, ни v, доказательство распадается на три случая.

Случай 1. u и v — оба QR. Тогда $u \equiv a^2 \pmod{p}$ и $v \equiv b^2 \pmod{p}$ для некоторых целых a и b. Отсюда следует, что $u \cdot v \equiv (a \cdot b)^2 \pmod{p}$, т.е. $u \cdot v$ — QR.

Случай 2. Ровно одно из чисел u и v является QR, скажем, u — QR, а v — NQR. Допустим, что также $u \cdot v$ — QR. Тогда существуют целые числа a и b, такие, что $u \equiv a^2 \pmod{p}$ и $u \cdot v \equiv b^2 \pmod{p}$. Так как $a \not\equiv 0 \pmod{p}$, получаем, что $v \equiv (b/a)^2 \pmod{p}$. Противоречие.

Случай 3. Как u, так и v — NQR. Из леммы А.11 мы знаем, что $i \cdot u, i = 1, 2, \ldots, p-1$, пробегает все ненулевые элементы по модулю p. Для $\frac{p-1}{2}$ значений i, для которых i — QR, мы имеем, согласно случаю 2, что $i \cdot u$ — NQR. А для прочих $\frac{p-1}{2}$ значений i, являющихся NQR, $i \cdot u$ будет QR. Поэтому $u \cdot v$ — QR.

■

Хотя следующая теорема нигде в этой книге не используется, мы приводим ее, поскольку она часто бывает нужна в родственных областях дискретной математики.

Теорема А.22
Пусть p — нечетное простое. Тогда для любого целого числа v

$$\sum_{u=o}^{p-1} \chi(u) \cdot \chi(u+v) = \begin{cases} p-1, & \text{если } p \text{ делит } v, \\ -1 & \text{в противном случае.} \end{cases}$$

Доказательство. Если p делит v, то утверждение тривиально. Если p не делит v, то в силу теорем А.21 и А.20 мы имеем

$$\sum\nolimits_{u=0}^{p-1} \chi(u)\chi(u+v) = \sum\nolimits_{u=1}^{p-1} \chi(u)\chi(u)\chi(1+v/u) =$$

$$= \sum\nolimits_{i=1}^{p-1} \chi(1+v/u) = \sum\nolimits_{w \neq 1} \chi(w) = -1 + \sum\nolimits_{w=0}^{p-1} \chi(w) = -1.$$

■

Пусть u — QR, скажем, $u \equiv a^2 \pmod{p}$. По теореме Ферма $u^{(p-1)/2} \equiv a^{p-1} \equiv 1 \pmod{p}$. Таким образом, $\frac{p-1}{2}$ QR суть нули многочлена $x^{(p-1)/2} - 1$ над $\mathrm{GF}(p)$. Так как многочлен степени $(p-1)/2$ над $\mathrm{GF}(p)$ имеет самое большее $(p-1)/2$ нулей в $\mathrm{GF}(p)$ (см. теорему В.15), в $\mathrm{GF}(p)$

$$x^{(p-1)/2} = \prod\nolimits_{u-\mathrm{QR}} (x-u). \qquad (A.16)$$

Отсюда также следует, что $u^{(p-1)/2} \neq 1$, если u — NQR. Так как $(u^{(p-1)/2})^2 \equiv 1 \pmod{p}$ по теореме Ферма и так как сравнение $y^2 \equiv 1 \pmod{p}$ имеет только корни 1 и -1, получаем, что $u^{(p-1)/2} \equiv -1 \pmod{p}$, если u — NQR. Это доказывает следующую теорему для всех u, взаимно простых с p. Для случая $p|u$ теорема верна тривиальным образом.

Теорема А.23
Пусть p — нечетное простое. Тогда для всех целых чисел u

$$\left(\frac{u}{p}\right) \equiv u^{(p-1)/2} \pmod{p}.$$

Следствие А.24
Пусть p — нечетное простое. Тогда

$$\left(\frac{-1}{p}\right) = \begin{cases} +1, & \text{если } p \equiv 1 \pmod 4, \\ -1, & \text{если } p \equiv 3 \pmod 4. \end{cases}$$

Доказательство. $(-1)^{(p-1)/2} = 1$ тогда и только тогда, когда $p \equiv 1 \pmod 4$.

■

Еще одно значение символа Лежандра, которое понадобится нам позже, — это $(\frac{2}{p})$.

Теорема А.25
Пусть p — нечетное простое. Тогда

$$\left(\frac{2}{p}\right) = \begin{cases} +1, & \text{если } p \equiv \pm 1 \pmod 8, \\ -1, & \text{если } p \equiv \pm 3 \pmod 8. \end{cases}$$

Доказательство.

$$2^{\frac{p-1}{2}} \prod_{k=1}^{\frac{p-1}{2}} k \equiv \prod_{k=1}^{\frac{p-1}{2}} (2k) \equiv \left(\prod_{k=1}^{\lfloor \frac{p-1}{4} \rfloor} (2k)\right) \cdot \left(\prod_{k=1+\lfloor \frac{p-1}{2} \rfloor}^{\frac{p-1}{4}} (2k)\right) \equiv$$

$$\equiv (-1)^{\frac{p-1}{2} - \lfloor \frac{p-1}{4} \rfloor} \cdot \left(\prod_{k=1}^{\lfloor \frac{p-1}{4} \rfloor} (2k)\right) \cdot \left(\prod_{k=1+\lfloor \frac{p-1}{4} \rfloor}^{\frac{p-1}{2}} (p-2k)\right) \equiv$$

$$\equiv (-1)^{\frac{p-1}{2} - \lfloor \frac{p-1}{4} \rfloor} \cdot \left(\prod_{k=1}^{\frac{p-1}{2}} (k)\right) \pmod{p}.$$

Деля обе части приведенного сравнения на $\prod_{k=1}^{\frac{p-1}{2}} k$, получаем

$$2^{\frac{p-1}{2}} \equiv (-1)^{\frac{p-1}{2} - \lfloor \frac{p-1}{4} \rfloor} \pmod{p}.$$

Теперь наше утверждение следует из теоремы А.23.

■

Напомним определение символа Якоби в терминах символа Лежандра:

$$\left(\frac{u}{m}\right) = \prod_i \left(\frac{u}{p_i}\right)^{e_i}, \qquad \text{где } m = \prod_i p_i^{e_i}. \tag{A.17}$$

Теорема А.26
Пусть m и n — нечетные целые числа. Тогда для символа Якоби выполняются следующие соотношения:

i) $\left(\frac{u}{m}\right) = \left(\frac{u-m}{m}\right)$,
ii) $\left(\frac{uv}{m}\right) = \left(\frac{u}{m}\right)\left(\frac{v}{m}\right)$,
iii) $\left(\frac{u}{mn}\right) = \left(\frac{u}{m}\right)\left(\frac{u}{n}\right)$,
iv) $\left(\frac{-1}{m}\right) = 1$ тогда и только тогда, когда $m \equiv 1 \pmod 4$,
v) $\left(\frac{2}{m}\right) = 1$ тогда и только тогда, когда $m \equiv \pm 1 \pmod 8$.

Доказательство. Первые два соотношения выполняются для символа Лежандра и, в силу (А.17), также для символа Якоби. Третье соотношение — прямое следствие из (А.17).

Чтобы увидеть, как четвертое соотношение вытекает из (А.17) и следствия А.24, достаточно заметить, что произведение нечетного числа целых чисел, сравнимых с 3 по модулю 4, также сравнимо с 3 по модулю 4, тогда как произведение четного числа таких чисел будет сравнимо с 1 по модулю 4. Доказательство последнего соотношения аналогично (используйте теорему А.25).

■

Для быстрого вычисления $\left(\frac{u}{m}\right)$ требуется еще одно соотношение. Мы не даем его доказательства, потому что соответствующая теория лежит за пределами этой книги. Заинтересованный читатель отсылается к теореме 99 в [HarW45] или к теореме 7.2.1 в [Shap83][4].

Теорема А.27 (*закон Гаусса квадратичной взаимности*)
Пусть m и n — нечетные взаимно простые целые числа. Тогда

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{(m-1)(n-1)}{4}}.$$

Соотношения из теорем А.25, А.26 и А.27 позволяют очень быстро вычислять символ Якоби.

Пример А.4

$$\left(\frac{12703}{16361}\right) \overset{A.27}{=} \left(\frac{16361}{12703}\right) \overset{A.26i)}{=} \left(\frac{3658}{12703}\right) \overset{A.26ii)}{=}$$

$$\overset{A.26ii)}{=} \left(\frac{2}{12703}\right) \cdot \left(\frac{1829}{12703}\right) \overset{A.26v)\&A.27}{=} \left(\frac{12703}{1829}\right) \overset{A.26i)}{=} \left(\frac{1729}{1829}\right) \overset{A.27}{=}$$

$$\overset{A.27}{=} \left(\frac{1829}{1729}\right) \overset{A.26i)}{=} \left(\frac{100}{1729}\right) \overset{A.26ii)}{=} \left(\frac{2}{1729}\right)^2 \cdot \left(\frac{5}{1729}\right)^2 = 1.$$

[4] На русском языке доказательство можно найти, например, в книгах *[АйеР87], *[Серр72], *[Кобл01], *[Чер02] и многих других. — *Прим. ред.*

Читателю нетрудно проверить, что описанный выше метод имеет приблизительно ту же сложность, что и алгоритм Эвклида.

Конечно, мы можем вычислить $(\frac{12703}{16361})$ прямо в пакете "Mathematica", как это было показано раньше.

```
JacobiSymbol[12703, 16361]
```

```
1
```

A.5 Непрерывные дроби

Довольно часто желательно аппроксимировать действительное число рациональным числом. Например, многие люди используют 22/7 в качестве приближения к π. Лучшее приближение к π дает дробь 333/106, а еще лучшее — 355/113. Можно увеличить знаменатель до 33102, чтобы получить еще большее улучшение.

```
N[Pi - 22 / 7]
N[Pi - 333 / 106]
N[Pi - 355 / 113]
N[Pi - 103993 / 33102]
```

```
-0.00126449
```

```
0.0000832196
```

$$-2.66764\times10^{-7}$$

$$5.77891\times10^{-10}$$

Теория непрерывных дробей объясняет, как получать такие хорошие приближения.

Определение A.12
Конечная непрерывная (или *цепная*) *дробь* — это выражение вида

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ldots + \cfrac{1}{a_{m-1} + \cfrac{1}{a_m}}}}}, \qquad (A.18)$$

где $a_0 \in \mathbb{Z}$ и $a_i \in \mathbb{N}, 1 \leq i \leq m$. Такая дробь часто обозначается последовательностью $[a_0, a_1, \ldots, a_m]$.

Если $m \to \infty$, мы говорим о *бесконечной* непрерывной дроби. Она имеет вид

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\dots}}}}$$

и записывается посредством $[a_0, a_1, a_2, \dots]$.

Очевидно, каждая конечная непрерывная дробь представляет рациональное число. Его можно найти, шаг за шагом упрощая непрерывную дробь, начиная с $a_{m-1} + \frac{1}{a_m} = \frac{a_{m-1}a_m + 1}{a_m}$, $\cfrac{1}{a_{m-1} + \cfrac{1}{a_m}} = \frac{a_m}{a_{m-1}a_m + 1}$ и т.д.

В пакете "Mathematica" это можно сделать с помощью функции *Normal*.

```
Normal[3 + 1/(7 + 1/(15 + 1/(1 + 1/292))]
```

$$\| \quad \frac{103993}{33102}$$

Покажем теперь, что верно и обратное: каждое рациональное число представляется конечной непрерывной дробью.

Лемма А.28
Каждое рациональное число имеет конечную непрерывную дробь.

Доказательство. Пусть $a/b, b > 0$ — рациональное число. Применим простую версию алгоритма Эвклида (алг. А.7) к паре (a, b). Таким образом, мы полагаем $s_0 = a, s_1 = b$ и рекурсивно вычисляем $s_i = q_i s_{i+1} + s_{i+2}$, где $0 \leq s_{i+2} < s_{i+1}$, пока не получим $s_{m+2} = 0$ (и потому $s_m = q_m s_{m+1}$) для некоторого целого m. Тогда

$$\frac{a}{b} = \frac{s_0}{s_1} = \frac{q_0 s_1 + s_2}{s_1} = q_0 + \frac{1}{s_1/s_2} = q_0 + \frac{1}{(q_1 s_2 + s_3)/s_2} =$$

$$= q_0 + \frac{1}{q_1 + 1/(s_2/s_3)} = \dots = q_0 + \cfrac{1}{q_1 + \cfrac{1}{\dots + \cfrac{1}{\dots + \cfrac{1}{q_{m-1} + \cfrac{1}{s_m/s_{m+1}}}}}} =$$

$$= q_0 + \cfrac{1}{q_1 + \cfrac{1}{\ldots + \cfrac{1}{\ldots q_{m-1} + \cfrac{1}{q_m}}}}.$$

Мы заключаем, что a/b имеет $[q_0, q_1, \ldots, q_m]$ своей непрерывной дробью.

■

Важно отметить, что представление рационального числа в виде конечной непрерывной дроби, где все q_i $(i \geq 1)$ положительны, не вполне однозначно. Хотя способ, при котором q_i вычисляются с помощью простой версии алгоритма Эвклида (см. доказательство выше), дает единственные значения всех q_i, ясно, что на последнем шаге мы всегда получаем $q_m \geq 2$, так как $s_{m+1} < s_m$.

В разложении положительного числа, не равного единице, мы можем переписать последний член следующим образом:

$$\frac{1}{q_m} = \frac{1}{(q_m - 1) + \frac{1}{1}}.$$

Это показывает, что $[q_0, q_1, \ldots, q_m]$ имеет то же значение, что и $[q_0, q_1, \ldots, q_m - 1, 1]$.

Последний член в непрерывной дроби можно выбирать таким образом, чтобы сделать число членов в разложении либо четным, либо нечетным, если это может оказаться удобным.

Формула (A.18) предлагает следующий путь вычисления непрерывной дроби числа α.

Алгоритм A.29
Непрерывная дробь числа α может быть вычислена следующим образом:

initialize $\alpha_0 = \alpha$,
compute (рекурсивно) $a_i = \lfloor \alpha_i \rfloor$ и
$\alpha_{i+1} = 1/(\alpha_i - a_i)$ для $i \geq 0$,
output $[a_0, a_1, a_2, \ldots]$.

Пример A.5

Рассмотрим $\alpha = 11/9$. *Тогда получим*

```
Clear[a];
alpha = 11 / 9; alpha[0] = alpha;
a[0] = ⌊ alpha[0] ⌋
alpha[1] = 1 / (alpha[0] - a[0]);
```

```
1
```

Для получения следующего члена вычислим

```
a[1] = ⌊alpha[1]⌋
alpha[2] = 1 / (alpha[1] - a[1]);
```

```
4
```

Продолжаем далее:

```
a[2] = ⌊alpha[2]⌋
alpha[3] = 1 / (alpha[2] - a[2]);
```

```
2
```

Power :: infty : Infinite expression $\frac{1}{0}$ *encountered.*

Мы заключаем, что $\alpha_2 = a_2$ *и что эта непрерывная дробь задается последовательностью* $[1, 4, 2]$. *Это можно легко проверить:*

```
Normal[1 + 1/(4 + 1/2)]
```

$$\frac{11}{9}$$

Чтобы вычислить в "Mathematica" непрерывную дробь числа, нужно сначала загрузить пакет *NumberTheory'ContinuedFractions'*.

```
<< NumberTheory`ContinuedFractions`
```

Для нахождения непрерывной дроби рационального числа можно использовать функцию *ContinuedFraction*.

```
ContinuedFraction[135 / 159]
```

$$0 + \cfrac{1}{1 + \cfrac{1}{5 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{2}}}}}}$$

Если α не рационально, можно включить число членов, которые желательно увидеть.

```
ContinuedFraction[Pi, 11]
```

$$3+\cfrac{1}{7+\cfrac{1}{15+\cfrac{1}{1+\cfrac{1}{292+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{1+\frac{1}{3}}}}}}}}}}$$

Чтобы выразить непрерывную дробь обычной дробью, можно снова воспользоваться функцией *Normal* из пакета "Mathematica".

```
Normal[ContinuedFraction[Pi, 11]]
```

$$\frac{4272943}{1360120}$$

Если непрерывная дробь задана в виде $[a_0, a_1, \ldots, a_m]$, то обычную непрерывную дробь можно получить с помощью функции *ContinuedFractionForm*. Читателю следует знать, что в "Mathematica" перечисление индексов начинается с 1, 2 и т.д.

```
AA = {3, 7, 15, 1, 292}
ContinuedFractionForm[AA]
```

$$3+\cfrac{1}{7+\cfrac{1}{15+\cfrac{1}{1+\frac{1}{292}}}}$$

Чтобы получить непрерывную дробь числа α в виде $[a_0, a_1, \ldots, a_m]$, можно просто добавить [[1]] к функции ContinuedFraction$[\alpha, m]$.

```
ContinuedFraction[Pi, 11][[1]]
```

```
{3, 7, 15, 1, 292, 1, 1, 1, 2, 1}
```

Определение А.13
k-я подходящая дробь C_k непрерывной дроби $[a_0, a_1, \ldots, a_m]$, $0 \leq$ $\leq k \leq m$, дается выражением $[a_0, a_1, \ldots, a_k]$.

Эти подходящие дроби можно легко вычислять с помощью функций *Table*, *Normal*, *Take*, *ContinuedFractionForm* и *Length*.

```
AA = {3, 7, 15, 1, 292}
Table[Normal[ContinuedFractionForm[Take[AA, i]]],
    {i, 1, Length[AA]}]
```

$$\left\{3, \frac{22}{7}, \frac{333}{106}, \frac{355}{113}, \frac{103993}{33102}\right\}$$

Каждая подходящая дробь, будучи рациональным числом, может быть записана в виде p_k/q_k. Значения p_k и q_k можно найти с помощью функций *Numerator* и *Denominator* из пакета "Mathematica".

```
C5 = Normal[ContinuedFraction[Pi, 5]]
p5 = Numerator[C5]
q5 = Denominator[C5]
```

$$\frac{103993}{33102}$$

103993

33102

Следующая теорема дает важные соотношения между непрерывной дробью и ее подходящими дробями. Чтобы сделать доказательство более коротким, мы ослабим наше обычное ограничение целости a_i.

Теорема А.30
Пусть $\{a_i\}_{i\geq 0}$ — конечная или бесконечная последовательность действительных чисел, все из которых положительны с возможным исключением для a_0.
Пусть $C_k = p_k/q_k$ определяется выражением $[a_0, a_1, \ldots, a_k]$, как в (А.18). Тогда числа p_k и q_k удовлетворяют рекуррентным соотношениям

$$\begin{array}{ll} p_0 = a_0, & p_1 = a_0a_1 + 1, \\ q_0 = 1, & q_1 = a_1, \\ p_k = a_kp_{k-1} + p_{k-2}, & k \geq 2, \\ q_k = a_kq_{k-1} + q_{k-2}, & k \geq 2. \end{array}$$

Доказательство. Воспользуемся индукцией по k. Для $k = 0$ мы имеем $p_0/q_0 = C_0 = a_0 = a_0/1$, так что в самом деле $p_0 = a_0$ и $q_0 = 1$.

Для $k = 1$ имеем $p_1/q_1 = C_1 = [a_0, a_1] = a_0 + 1/a_1 = (a_0a_1 + 1)/a_1$, так что в самом деле $p_1 = a_0a_1 + 1$ и $q_1 = a_1$.

Допустим, что теорема уже доказана для некоторого значения k. Таким образом,

$$C_k = [a_0, a_1, \ldots, a_k] = \frac{p_k}{q_k} = \frac{a_kp_{k-1} + p_{k-2}}{a_kq_{k-1} + q_{k-2}}.$$

Сделаем теперь замену $a_k \mapsto a_k + 1/a_{k+1}$. Тогда

$$C_{k+1} \overset{\text{Опр.}}{=} [a_0, a_1, \ldots, a_k, a_{k+1}] \overset{\text{Опр.А.12}}{=} [a_1, a_2, \ldots, a_k + \frac{1}{a_{k+1}}] =$$

$$\overset{\text{индук.}}{=} \frac{(a_k + \frac{1}{a_{k+1}})p_{k-1} + p_{k-2}}{(a_k + \frac{1}{a_{k+1}})q_{k-1} + q_{k-2}} = \frac{a_{k+1}(a_k p_{k-1} + p_{k-2}) + p_{k-1}}{a_{k+1}(a_k q_{k-1} + q_{k-2}) + q_{k-1}} =$$

$$\overset{\text{рек.соотн.}}{=} \frac{a_{k+1}p_k + p_{k-1}}{a_{k+1}q_k + q_{k-1}} \overset{\text{рек.соотн.}}{=} \frac{p_{k+1}}{q_{k+1}}.$$

■

Небольшим результатом, который понадобится нам позже, служит неравенство

$$q_k \geq F_k, \tag{A.19}$$

где F_k — k-е число Фибоначчи, последовательность которых определяется равенствами $F_0 = 0, F_1 = 1$ и рекуррентным соотношением $F_k = F_{k-1} + F_{k-2}, k \geq 2$. С помощью легкого индуктивного рассуждения неравенство $q_k \geq F_k$ получается из неравенств $q_0 > 0, q_1 \geq 1$ и рекуррентного соотношения $q_k = a_k q_{k-1} + q_{k-2}$, в котором $a_k \geq 1$ (используйте то, что $q_k \geq q_{k-1} + q_{k-2}$).

Лемма А.31
Пусть $C_k = p_k/q_k$ — k-я подходящая дробь некоторой непрерывной дроби. Тогда

$$p_k q_{k-1} - p_{k-1}q_k = (-1)^{k-1}.$$

Доказательство. Снова воспользуемся индукцией по k. При $k = 1$ по теореме А.30 получаем $p_1 q_0 - p_0 q_1 = (a_0 a_1 + 1) \times 1 - a_0 \times a_1 = 1$.

Чтобы доказать шаг индукции от k к $k+1$, используем рекуррентное соотношение из теоремы А.30:

$$p_{k+1}q_k - p_k q_{k+1} \overset{\text{А.30}}{=} (a_{k+1}p_k + p_{k-1})q_k - p_k(a_{k+1}q_k + q_{k-1}) =$$

$$= p_{k-1}q_k - p_k q_{k-1} \overset{\text{инд.}}{=} (-1)(-1)^{k-1} = (-1)^k.$$

■

Следствие А.32
Пусть $C_k = p_k/q_k$ — k-я подходящая дробь некоторой непрерывной дроби. Тогда

$$\text{НОД}(p_k, q_k) = 1.$$

Доказательство. Это — непосредственное следствие равенства $p_{k-1}q_k - p_k q_{k-1} = (-1)^{k-1}$. Действительно, каждое число, делящее p_k и q_k, должно делить -1.

■

Теорема А.33
Пусть $C_k = p_k/q_k$ — k-я подходящая дробь некоторой конечной или бесконечной непрерывной дроби $[a_0, a_1, \ldots]$. Тогда

$$C_k - C_{k-1} = \frac{(-1)^{k-1}}{q_{k-1}q_k}, \qquad k \geq 1, \tag{A.20}$$

$$C_k - C_{k-2} = \frac{a_k(-1)^k}{q_{k-2}q_k}, \qquad k \geq 2, \tag{A.21}$$

$$C_0 < C_2 < C_4 < \ldots < C_5 < C_3 < C_1. \tag{A.22}$$

Для бесконечной непрерывной дроби строго возрастающая ограниченная последовательность подходящих дробей с четными номерами имеет тот же предел, что и строго убывающая ограниченная последовательность подходящих дробей с нечетными номерами.

Доказательство. По лемме А.31 и теореме А.30

$$C_k - C_{k-1} = \frac{p_k}{q_k} - \frac{p_{k-1}}{q_{k-1}} = \frac{p_kq_{k-1} - p_{k-1}q_k}{q_{k-1}q_k} = \frac{(-1)^{k-1}}{q_kq_{k-1}};$$

$$C_k - C_{k-2} = \frac{p_k}{q_k} - \frac{p_{k-2}}{q_{k-2}} = \frac{p_kq_{k-2} - p_{k-2}q_k}{q_{k-2}q_k} =$$

$$= \frac{(a_kp_{k-1} + p_{k-2})q_{k-2} - p_{k-2}(a_kq_{k-1} + q_{k-2})}{q_{k-2}q_k} =$$

$$= \frac{a_kp_{k-1}q_{k-2} - a_kp_{k-2}q_{k-1}}{q_{k-2}q_k} = \frac{a_k(-1)^k}{q_kq_{k-2}}.$$

Это доказывает соотношения А.20 и А.21. То, что подходящие дроби с четными номерами образуют строго возрастающую последовательность, следует из равенства (А.21), влекущего, что $C_{2k} - C_{2k-2} > 0$ (где все a_i положительны). По той же самой причине подходящие дроби с нечетными номерами строго убывают.

Для доказательства того, что каждая подходящая дробь с четным номером, скажем, C_{2i}, меньше, чем любая подходящая дробь с нечетным номером, скажем, C_{2j+1}, заметим сначала,что $C_{2k+1} - C_{2k} > 0$ в силу (А.20). Комбинируем это с вышесказанным и получаем

$$C_{2i} < C_{2i+2j} < C_{2i+2j+1} < C_{2j+1}.$$

Наконец, по формулам (А.19) и (А.20) для $k \geq 2$

$$|C_k - C_{k-1}| = \frac{1}{q_{k-1}q_k} \leq \frac{1}{F_{k-1}F_k} \leq \frac{1}{(k-1)^2},$$

и потому разность между двумя последовательными подходящими дробями стремится к нулю, когда k стремится к бесконечности. Это показывает, что предел подходящих дробей с четными номерами должен совпадать с пределом подходящих дробей с нечетными номерами.

■

Пример A.6

Ниже мы выписываем первые 10 *подходящих дробей числа* π *в их естественном порядке.*

```
<< NumberTheory`ContinuedFractions`
```

```
Do[Print[k - 1, " ",
  N[Normal[ContinuedFraction[Pi, k]], 16]], {k, 1, 9, 2}]
Print[π, " ", N[π, 16]]
Do[Print[k - 1, " ",
  N[Normal[ContinuedFraction[Pi, k]], 16]], {k, 10, 2, -2}]
```

```
0  3.
2  3.141509433962264
4  3.141592653011902
6  3.141592653467437
8  3.141592653581078
π  3.141592653589793
9  3.141592653591404
7  3.141592653618936
5  3.141592653921421
3  3.141592920353983
1  3.142857142857143
```

Следующие две теоремы формулируются без доказательств. Их можно найти в любом вводном курсе непрерывных дробей, например, в [Rose84], но рассуждения там слишком техничны для наших целей[5].

Теорема A.34
Пусть $C_k = p_k/q_k$ — k-ая подходящая дробь конечной или бесконечной непрерывной дроби $\alpha = [a_0, a_1, \ldots]$, и допустим, что $|\alpha - r/s| < |\alpha - p_k/q_k|$. Тогда $s > q_k$.

Например, поскольку 355/113 — подходящая дробь для π, мы теперь знаем, что ближе к π, чем 355/113, лежат только дроби со заменателем, большим 113.

[5] Доказательства на русском языке можно найти в ряде книг. См., например, *[Хинч78] или *[Ленг70]. — *Прим. ред.*

Теорема А.35
Пусть $\alpha \in \mathbb{R}$ и r/s (с НОД$(r,s) = 1$) — такая рациональная дробь, что $|\alpha - r/s| < 1/2s^2$.
Тогда r/s — подходящая дробь к разложению α в непрерывную дробь.

Эта теорема говорит, что рациональное число r/s, которое лежит на расстоянии, не превосходящем $1/2s^2$, от числа α, оказывается подходящей дробью для разложения этого числа.

А.6 Формула обращения Мёбиуса, принцип включения и исключения

А.6.1 Формула обращения Мёбиуса

В дискретной математике некоторая функция f часто определяется в терминах другой функции, скажем, g. Задача состоит в том, чтобы выразить функцию g в терминах f. Часто эту задачу можно решить с помощью теории частично упорядоченных множеств и (обобщенной) формулы обращения Мёбиуса (см. гл. 4 в [Aign79]). В данном разделе мы обсудим два важных частных случая. Оба они рассматриваются в упомянутой выше теории, но оказывается, что они могут быть доказаны непосредственно.

Мы часто будем нуждаться в явном примарном разложении целого числа n. В дальнейшем мы не придерживаемся строгого упорядочения простых чисел, задаваемого равенствами $p_1 = 2, p_2 = 3$ и т.д. Однако различным индексам все еще отвечают различные простые числа.

Определение А.14
Пусть $n = \prod_{i=1}^{k} p_i^{e_i}, e_i > 0, 1 \le i \le k$, где все p_i — различные простые числа. Тогда *функция Мёбиуса* $\mu(n)$ определяется равенством

$$\mu(n) = \begin{cases} 1, & \text{если } n = 1, \\ 0, & \text{если } e_i \ge 2 \text{ для некоторого } i, 1 \le i \le k, \\ (-1)^k, & \text{если все } e_i \text{ равны 1.} \end{cases}$$

Другими словами, $\mu(n)$ — мультипликативная функция, удовлетворяющая равенствам $\mu(1) = 1, \mu(p) = -1$ и $\mu(p^i) = 0, i \ge 2$, для любого простого p. "Mathematica" содержит стандартную функцию *MoebiusMu*, вычисляющую μ.

```
n = 30; MoebiusMu[n]
```

```
-1
```

Функция Мёбиуса, определенная таким путем, обладает следующим свойством.

Лемма A.36
Пусть n — положительное целое число. Тогда

$$\sum_{d|n}\mu(d)=\begin{cases}1, & \text{если } n=1,\\ 0, & \text{если } n>1.\end{cases}$$

Доказательство. При $n=1$ утверждение тривиально. Для $n>1$ запишем, как выше, $n=\prod_{i=1}^{k}p_i^{e_i}, e_i>0, 1\le i\le k$. Тогда $k>0$ и поэтому

$$\sum\nolimits_{d|n}\mu(d)=\sum\nolimits_{d|p_1^{e_1}p_2^{e_2}\ldots p_k^{e_k}}\mu(d)=$$

$$=1+\sum\nolimits_{l=1}^{k}\sum\nolimits_{1\le i_1<i_2<\ldots<i_l\le k}\mu(p_{i_1}p_{i_2}\ldots p_{i_l})=\sum\nolimits_{l=0}^{k}\binom{k}{l}(-1)^l=(1-1)^k.$$

■

Читатель может проверить эту лемму посредством следующих вычислений:

```
DivisorSum[f_, m_] := Plus @@ (f /@ Divisors[m])
```

```
m = 100; DivisorSum[MoebiusMu[m]]
```

```
0
```

Лемма A.37
Пусть m и n — такие натуральные числа, что m делит n. Тогда

$$\sum\nolimits_{d,\,m|d|n}\mu(n/d)=\begin{cases}1, & \text{если } m=n,\\ 0 & \text{в противном случае.}\end{cases}$$

Доказательство. Пусть $n=n'm$. Для каждого d, такого, что $m|d|n$, запишем $d=d'm$. Тогда $\sum_{d,\,m|d|n}\mu(n/d)=\sum_{d'|n'}\mu(n'/d')$, что по лемме A.36 равно 1, если $n'=1$ (т.е. $m=n$), и равно 0 при $n'>1$.

■

Теорема A.38 (*формула обращения Мёбиуса*)
Пусть f — функция[6], заданная на множестве $\mathbb{N}$, а функция g определена на $\mathbb{N}$ равенством

$$g(n)=\sum\nolimits_{d|n}f(d),\qquad n\in\mathbb{N}.$$

Тогда для всех $n\in\mathbb{N}$

$$f(n)=\sum\nolimits_{d|n}\mu(d)g(n/d).$$

[6] принимающая значения в мультипликативной абелевой группе. — *Прим. ред.*

Доказательство. В силу определения $g(n)$ и леммы А.37

$$\sum_{d|n} \mu(n/d)g(d) = \sum_{d|n} \mu(n/d) \sum_{e|d} f(e) =$$

$$= \sum_{e|n} f(e) \sum_{d,\, e|d|n} \mu(n/d) = f(n).$$

■

Следствие А.39 (*мультипликативная формула обращения Мёбиуса*)
Пусть F — функция, заданная на множестве $\mathbb{N}$ а функция G определена на $\mathbb{N}$ равенством

$$G(n) = \prod_{d|n} F(d), \qquad n \in \mathbb{N}.$$

Тогда для всех $n \in \mathbb{N}$

$$F(n) = \prod_{d|n} \mu(d)G(n/d) = \prod_{d|n} \mu(n/d)G(d).$$

Доказательство. Подставляем $g(n) = \log G(n)$ и $f(n) = \log F(n)$ в формулу обращения Мёбиуса.

■

Пример А.7

Теорема А.12 показывает, что функция Эйлера удовлетворяет равенству

$$\sum_{d|n} \varphi(d) = n.$$

Из формулы обращения Мёбиуса (теорема А.38) следует, что для $n = \prod_{i=1}^{k} p_i^{e_i}, e_i > 0, 1 \le i \le k$,

$$\begin{aligned} \varphi(n) &= \sum_{d|n} \mu(d)\tfrac{n}{d} = \\ &= \tfrac{n}{1} - \sum_{1\le i\le k} \tfrac{n}{p_i} + \sum_{1\le i<j\le k} \tfrac{n}{p_i p_j} - \ldots + (-1)^k \tfrac{n}{p_1 p_2 \ldots p_k} = \\ &= n(1-\tfrac{1}{p_1})(1-\tfrac{1}{p_2})\ldots(1-\tfrac{1}{p_k}). \end{aligned}$$

Это иным способом доказывает теорему А.17.

Теорема В.17 в разд. В.3 показывает важное применение мультипликативной формулы обращения Мёбиуса.

А.6.2 Принцип включения и исключения

Мы завершим этот раздел еще одним полезным принципом. Чтобы несколько стимулировать интуицию, рассмотрим целые числа между 0 и $p \cdot q - 1$, где p и q — различные простые числа. Мы хотим прямым способом вычислить $\varphi(p \cdot q)$, т.е. подсчитать количество целых чисел $i, 0 \le i < p \cdot q$,

которые взаимно просты с $p \cdot q$. Оно равно $p \cdot q$ минус количество целых чисел $i, 0 \leq i < p \cdot q$, которые имеют нетривиальные общие делители с $p \cdot q$, т.е. которые делятся на p или q. Имеется q кратных числа p в области $0, 1, \ldots, p \cdot q - 1$ и, аналогично, p кратных числа q. Однако есть одно число, которое кратно как p, так и q, это — нуль. Мы получаем, что

$$\varphi(p \cdot q) = p \cdot q - p - q + 1 = (p-1)(q-1) = p \cdot q \left(1 - \frac{1}{p}\right)\left(1 - \frac{1}{q}\right),$$

как и должно быть согласно теореме A.17.

Теорема A.40 (*принцип включения и исключения*)
Пусть S — конечное множество из N элементов. Предположим, что элементы из S могут удовлетворять определенным свойствам $P(i), 1 \leq i \leq k$.
Пусть $N(i_1, i_2, \ldots, i_s)$ — число элементов из S, которые удовлетворяют свойствам $P(i_1), P(i_2), \ldots, P(i_s)$, где $1 \leq i_1 < i_2 < \ldots < i_s \leq k, 1 \leq s \leq k$ (и возможно также некоторым другим свойствам).
Пусть $N(\emptyset)$ — число элементов из S, которые не удовлетворяют ни одному из свойств $P(i), 1 \leq i \leq k$.
Тогда

$$N(\emptyset) = N - \textstyle\sum_{1 \leq i \leq k} N(i) + \sum_{1 \leq i < j \leq k} N(i,j) - \ldots + \\ +(-1)^k N(1, 2, \ldots, k).$$

Доказательство. Элемент s из S, который удовлетворяет в точности r из k свойств, подсчитывается выражениями

$$1 - \binom{r}{1} + \binom{r}{2} - \ldots + (-1)^r \binom{r}{r} = (1-1)^r = \begin{cases} 1, & \text{если } r = 0, \\ 0, & \text{если } r \neq 0 \end{cases}$$

как в правой, так и в левой частях равенства.

■

Мы оставляем читателю в качестве упражнения вывод теоремы A.17 непосредственно из определения функции Эйлера и указанного выше принципа. (Совет: Пусть $p_i, 1 \leq i \leq k$, — все простые числа, которые делят n; возьмем $S = \{0, 1, \ldots, n-1\}$ и скажем, что элемент $s \in S$ обладает свойством $P(i), 1 \leq i \leq k$, если s делится на p_i.)

A.7 Задачи

Задача A.1^M. Пусть $\prod_{i=1}^{k} p_i^{a_i}$ — примарное разложение целого числа n. Сколько различных делителей имеет n? Проверьте свой ответ для $n = 1000$, используя функцию *DivisorSigma*[k,n], которая вычисляет $\sum_{d|n} d^k$ (возьмите $k = 0$).

Задача А.2^M. Вычислите такие u и v, что НОД(455,559) $= 455u + 559v$.

Задача А.3. Докажите, что НОД$(a^m - 1, a^n - 1) = a^{\text{НОД}(m,n)-1}$ для любого натурального числа a. (Совет: сведите пару $\{m, n\}, m \geq n$, к $\{m - n, n\}$, а затем следуйте простой версии алгоритма Эвклида.)

Задача А.4^M. a) Проверьте, что 563 — простое число.
b) Используйте алгоритм Эвклида, чтобы вычислить $11^{-1} \bmod 563$.
c) Решите сравнение $11x \equiv 85 \pmod{563}$.

Задача А.5. Найдите решения сравнения $33x \equiv 255 \pmod{1689}$. Заметьте, что $1689 = 4 \times 563$ и используйте результаты задачи А.4.

Задача А.6. a) Определите $\varphi(100)$. Проверьте результат посредством функции *EulerPhi*.
b) Вычислите две наименьших значащих цифры числа 2004^{2004} без использования компьютера.

Задача А.7^M. Решите систему сравнений (совет: используйте теорему А.19):

$$3x \equiv 2 \pmod{11}, \qquad 7x \equiv 8 \pmod{13}, \qquad 4x \equiv 14 \pmod{15}.$$

Задача А.8^M. Определите символ Якоби (7531/3465).

Задача А.9. Используйте китайскую теорему об остатках, чтобы решить сравнение $x^2 \equiv 56 \pmod{143}$. (Совет: сначала сведите его к некоторой системе линейных сравнений.) Сколько имеется различных решений по модулю 143?

Задача А.10. Определите первые пять членов непрерывной дроби числа f, наибольшего корня уравнения $f^2 = f + 1$. Определите также первые пять подходящих дробей.

Что вы думаете о других членах в непрерывной дроби числа f? Докажите свою гипотезу (совет: используйте алг. А.29 и определение f).

Задача А.11. Докажите теорему А.17 с помощью принципа включения и исключения (теорема А.40) и определения функции Эйлера φ.

Приложение B

Конечные поля

Вводные замечания

Большинство читателей знакомо с алгебраической структурой множеств рациональных, действительных и комплексных чисел. Эти множества обладают свойствами сложения и умножения, которые "желательны". Такие множества называются *полями*.

В дискретной математике, в частности, в криптологии и теории кодирования существенную роль играют поля конечной мощности. В этой главе дается введение в теорию конечных полей[1].

Укажем общие очертания этого приложения.

В разд. B.1 мы вводим основные определения и теоремы абстрактной алгебры и линейной алгебры. В частности, мы покажем, что множество целых чисел, взятых по модулю некоторого простого числа, образуют конечное поле. В разд. B.2 будет дана общая конструкция конечных полей. В разд. B.3 выводится формула для числа неприводимых многочленов над данным конечным полем. Это покажет, что конечные поля существуют размеров, равных любой степени простого числа. Анализ строения конечных полей будет дан в разд. B.4. В частности, будет показано, что конечное поле размера q существует тогда и только тогда, когда q — степень простого числа. Более того, такое поле единственно, его аддитивная группа имеет структуру векторного пространства, а мультипликативная группа [ненулевых элементов] циклична.

B.1 Алгебра

Хотя мы предполагаем, что читатель уже знаком с понятиями, обсуждаемыми в этом и следующем разделах, мы для удобства чтения предлагаем это резюме.

B.1.1 Абстрактная алгебра

□ Операции на множестве

Пусть S — непустое множество. Операция $*$, определенная на S, — это отображение из $S \times S$ в S. Образ пары (s, t) относительно $*$ обозначается

[1] На русском языке более детальное изложение можно найти в *[ЛидП96] и особенно в *[ЛидН88]. — *Прим. ред.*

через $s * t$. Примерами операций служат сложение "+" в $\mathbb{R}$ и умножение "$\times$" в $\mathbb{C}$. Операция $*$ называется коммутативной, если для любых s и t из S

S.1. $s * t = t * s$ для всех s и t из S.

Элемент e из S, удовлетворяющий равенствам

S.2. $s * e = e * s = s$ для всех s из S

называется единичным [нейтральным] элементом в $(S, *)$.

Единичный элемент в $(S, *)$ единственен. В самом деле, предположим, что как e, так и e' удовлетворяют условию S.2. Тогда, применяя S.2 дважды, получаем

$$e = e * e' = e'.$$

Пример В.1

Возьмем в качестве S множество целых чисел $\mathbb{Z}$ и сложение $+$ в качестве операции. Эта операция коммутативна, а 0 — единичный элемент в $(\mathbb{Z}, +)$.

Пример В.2

Пусть S — множество действительных 2×2-матриц с матричным умножением в качестве операции. Эта операция некоммутативна; например,

```
(1 1) * (0 1) == (0 1) * (1 1)
(0 1)   (1 1)    (1 1)   (0 1)
```

```
False
```

С другой стороны, это множество S обладает единичным элементом, именно $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Вычислим для примера

```
MatrixForm[(a b) * (1 0)]
          (c d)   (0 1)
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

□ Группа

Определение В.1
Пусть G — непустое множество, на котором определена операция $*$. Пара $(G, *)$ называется *группой*, если

G1. $(g*h)*k = g*(h*k)$ для всех $g, h, k \in G$ (ассоциативность);

G2. G содержит единичный элемент, скажем, e;

G3. для каждого $g \in G$ существует элемент $h \in G$, такой, что $g * h = h * g = e$; этот элемент называется *обратным* к g и обычно обозначается через g^{-1}.

Свойство G1 говорит, что нет нужды писать скобки в цепочках наподобие $g * h * k$. Элемент h в свойстве G3 определен однозначно. В самом деле, если как h, так и h' удовлетворяют G3, то $h = h * e = h * (g * h') = (h * g) * h' = e * h' = h'$. Тем же самым способом можно показать, что для каждых $a, b \in G$ уравнения

$$a \cdot x = b \qquad \text{и} \qquad x \cdot a = b$$

имеют единственные решения в G, а именно

$$x = a^{-1} \cdot b \qquad \text{и, соответственно,} \qquad x = b \cdot a^{-1}.$$

Читатель легко проверит, что $(\mathbb{Z}, +)$ из примера B.1 является коммутативной группой. Другие хорошо известные примеры коммутативных групп — это $(\mathbb{Q}, +)$, $(\mathbb{Q} \setminus \{0\}, \cdot)$ и $(\mathbb{R}, +)$.

Пример B.2 не дает группу, потому что не все матрицы имеют обратные (например, не имеет обратной матрица, состоящая из одних нулей).

Пусть $(G, *)$ — группа и H — подмножество из G с тем свойством, что $(H, *)$ — тоже группа; тогда H называется *подгруппой* в G. Можно показать (см. задачу B.3), что H — подгруппа в G тогда и только тогда, когда

$$h_1 h_2^{-1} \in G \qquad \text{для любых } h_1, h_2 \in G.$$

Пусть $m \in \mathbb{Z} \setminus \{0\}$ и $m\mathbb{Z} = \{mk | k \in \mathbb{Z}\}$. Тогда $(m\mathbb{Z}, +)$ — коммутативная подгруппа в $(\mathbb{Z}, +)$, как легко можно проверить.

Пример B.3

Пусть $m \in \mathbb{Z} \setminus \{0\}$ и $\mathbb{Z}_m^*$ — приведенная система вычетов:

$$\mathbb{Z}_m^* = \{i | 0 \le i < m, \text{НОД}(i, m) = 1\}.$$

Мощность множества $\mathbb{Z}_m^*$ равна $\varphi(m)$ по определению A.6.

Из леммы A.13 следует, что произведение двух элементов из $\mathbb{Z}_m^*$ всегда можно представить снова элементом из $\mathbb{Z}_m^*$. Очевидно, 1 — элемент в $\mathbb{Z}_m^*$, который является единичным элементом относительно умножения. Далее, согласно теореме A.18, каждый элемент в $\mathbb{Z}_m^*$ имеет мультипликативный обратный (заметим, что для $a \in \mathbb{Z}_m^*$ мы имеем $\text{НОД}(a, m) = 1$ и поэтому сравнение $ax \equiv 1 \pmod{m}$ имеет единственное решение).

Мы заключаем, что мультипликативная группа $(\mathbb{Z}_m^*, \times)$ является коммутативной группой мощности $\varphi(m)$.

Коммутативные группы называют также *абелевыми* группами. Обычно абелевы группы представляются аддитивно: операция обозначается знаком "плюс", единичный элемент называется нулевым элементом (и обозначается нулем). В этих обозначениях абелева группа именуется аддитивной группой.

В этом введении наиболее часто используемые абелевы группы — это $(\mathbb{Z}_m, +)$, но в гл. 10 представлен также другой пример (см. теорему 10.2).

Рассмотрим также более интересную ситуацию, когда на множестве определены две операции. Действие первой мы обозначим через $g + h$, а второй — $g \cdot h$.

□ Кольцо

Определение В.2
Тройка $(R, +, \cdot)$ называется *кольцом*, если

R1. $(R, +)$ — коммутативная группа; ее единичный элемент обозначается 0;

R2. Операция $\cdot$ ассоциативна;

R3. Выполняется дистрибутивность, т.е. для всех $r, s, t \in R$ $r \cdot (s + t) = r \cdot s + r \cdot t$ и $(r + s) \cdot t = r \cdot t + s \cdot t$.

Начиная с этого момента, мы часто будем писать gh вместо $g \cdot h$. Аддитивное обратное к элементу g в группе $(R, +)$ будет обозначаться просто через $-g$, также мы пишем $2g$ вместо $g + g$, $3g$ вместо $g + g + g$ и т.д. Заметим, что 0 действительно ведет себя подобно нулевому элементу, поскольку для любого $r \in R$ мы имеем $0r = (r - r)r = r^2 - r^2 = 0$ и аналогичное равенство $r0 = 0$.

Допустим, что на $R \setminus \{0\}$ операция $\cdot$ коммутативна. Тогда кольцо $(R, +, \cdot)$ называется *коммутативным*. Примерами коммутативных колец служат $(\mathbb{R}, +, \cdot)$, $(\mathbb{Q}, +, \cdot)$, $(\mathbb{Z}, +, \cdot)$, а также $(m\mathbb{Z}, +, \cdot)$, когда m отлично от нуля.

Пусть $(R, +, \cdot)$ — кольцо, и S — подмножество из R с тем свойством, что $(S, +, \cdot)$ само является кольцом; тогда S называется *подкольцом* в R. Заметим, что $(6\mathbb{Z}, +, \cdot)$ — подкольцо в $(2\mathbb{Z}, +, \cdot)$, которое, в свою очередь, является подкольцом в $(\mathbb{Z}, +, \cdot)$.

□ Идеал

Определение В.3
Подкольцо $(S, +, \cdot)$ кольца $(R, +, \cdot)$ называется *идеалом*, если
I. Для всех $r \in R$ и $s \in S$ ($rs \in S$ и $sr \in S$).

Пусть $m \in \mathbb{Z} \setminus \{0\}$. Легко проверить, что любое целочисленное кратное числа, кратного m, снова кратно m. Отсюда следует, что $(m\mathbb{Z}, +, \cdot)$ — идеал в $(\mathbb{Z}, +, \cdot)$.

Допустим теперь, что кольцо $(R, \cdot)$ обладает единичным элементом, скажем, e, тогда некоторые элементы из R могут обладать обратными из R. Это такие $a \in R$, для которых существуют элементы $b \in R$, обладающие свойствами $ab = ba = e$. Такой обратный к a, который единственен, называется *мультипликативным обратным* и обозначается a^{-1}. Очевидно, элемент 0 не имеет мультипликативного обратного. Действительно, допустим, что $r0 = e$ для некоторого $r \in R$. Тогда для каждого $a \in R$ получим, что $a = ae = a(r0) = (ar)0 = 0$, т.е. $R = \{0\}$.

Из вышесказанного следует, что $(R, \cdot)$, когда $R \neq \{0\}$, не может быть группой. Однако $(R \setminus \{0\}, \cdot)$ вполне может обладать структурой группы.

□ Поле

Определение B.4
Тройка $(F, +, \cdot)$ называется *полем*, если

F1. $(F, +)$ — коммутативная группа; ее единичный элемент обозначается через 0;

F2. $(F \setminus \{0\}, \cdot)$ — группа; мультипликативный единичный элемент обозначается через e;

F3. Выполняется дистрибутивность.

В отличие от некоторых колец поле не может содержать ненулевых делителей нуля, т.е. ненулевых элементов f и g, чье произведение fg равно нулю. В самом деле, допустим, что $fg = 0$ и $f \neq 0$. Тогда $g = eg = (f^{-1}f)g = f^{-1}(fg) = f^{-1}0 = 0$; поэтому любой элемент из F равен нулю.

Если подкольцо $(K, +, \cdot)$ поля $(F, +, \cdot)$ имеет структуру поля, то мы будем называть его *подполем* поля $(F, +, \cdot)$.

Примерами служат поля рациональных чисел $(\mathbb{Q}, +, \cdot)$, действительных чисел $(\mathbb{R}, +, \cdot)$ и комплексных чисел $(\mathbb{C}, +, \cdot)$, каждое из которых — подполе следующего.

Мы говорим о конечных группе $(G, *)$, кольце $(R, +, \cdot)$ или поле $(F, +, \cdot)$ порядка n, если G, соответственно, R, F — конечное множество мощности n. Обычно мощность конечного поля обозначается буквой q.

В этой главе мы изучим структуру конечных полей. Оказывается, конечное поле порядка q существует только тогда, когда q — степень простого числа. Более того, для такого фиксированного q это поле по существу единственно, что оправдывает общепринятые обозначения $\mathbb{F}_q$ и $\mathrm{GF}(q)$ (где GF означает Galois Field — поле Галуа). Примеры конечных полей будут даны в разд. B.2.

Аналогично коммутативным кольцам мы определяем коммутативное поле $(F, +, \cdot)$ как поле, в котором группа $(F \setminus \{0\}, \cdot)$ коммутативна. Хотя следующая теорема очень важна, она здесь не доказывается (см. [Cohn77][2]).

[2]На русском языке доказательство можно найти во многих книгах; например, см.

Теорема B.1 (*Веддербарн*)
Любое конечное поле коммутативно.

□ Отношения эквивалентности

Определение B.5
Пусть U — множество. Для произвольного подмножества P из $U \times U$ определим на U *отношение* $\sim$ следующим условием: для любых $u, v \in U$

$$u \sim v \Longleftrightarrow (u, v) \in P.$$

Отношение эквивалентности есть отношение с дополнительными свойствами:

E1. Для всех $u \in U (u \sim u)$ (рефлексивность);

E2. Для всех $u, v \in U (u \sim v \Longrightarrow v \sim u)$ (симметричность);

E3. Для всех $u, v, w \in U [(u \sim v \& v \sim w) \Longrightarrow u \sim w]$ (транзитивность).

Пусть U — множество прямых линий в эвклидовой плоскости. Тогда свойство "быть параллельными или равными" определяет отношение эквивалентности.

В разд. A.3 мы видели другой пример. Там $U = \mathbb{Z}$ и для фиксированного $m, m \neq 0$, отношение $\equiv$ определяется условием "$a \equiv b \pmod{m}$ тогда и только тогда, когда m делит $a - b$".

Пусть $\sim$ — отношение эквивалентности, определенное на множестве U. Непустое подмножество W из U называется *классом эквивалентности*, если

E1) $\forall v, w \in W (v \sim w)$,

E2) $\forall w \in W \forall u \in U \setminus W (u \not\sim w)$.

Из указанных выше свойств следует, что класс эквивалентности состоит из всех элементов из U, находящихся в отношении $\sim$ с некоторым фиксированным элементом из U. Очевидно, различные классы эквивалентности в U образуют разбиение U. Класс эквивалентности, содержащий данный элемент w, будет обозначаться через $< w >$.

Пусть $(R, +, \cdot)$ — коммутативное кольцо с (мультипликативным) единичным элементом e и пусть $(S, +, \cdot)$ — идеал в $(R, +, \cdot)$. Определим отношение $\equiv$ на R условием

$$a \equiv b \pmod{S} \Longleftrightarrow a - b \in S. \qquad (B.1)$$

*[Холл62], где доказывается и более сильное утверждение об альтернативных кольцах (т.е. кольцах с ослабленным условием ассоциативности). — *Прим. ред.*

Читатель с легкостью проверит, что условие (B.1) определяет отношение эквивалентности. Пусть R/S (читается "R по модулю S") обозначает множество классов эквивалентности. Определим на R/S две операции следующими формулами:

$$< a > + < b >:=< a + b >, \qquad a, b \in R,$$

$$< a > \cdot < b >:=< ab >, \qquad a, b \in R.$$

Легко проверить, что эти определения не зависят от частного выбора a и b в классах эквивалентности $< a >$ и $< b >$ соответственно. В качестве упражнения мы оставляем читателю доказательство следующей теоремы.

Теорема B.2
Пусть $(R, +, \cdot)$ — коммутативное кольцо, а $(S, +, \cdot)$ — идеал в $(R, +, \cdot)$. При данном выше определении $(R/S, +, \cdot)$ — коммутативное кольцо с единичным элементом.

Кольцо $(R/S, +, \cdot)$ называется кольцом классов вычетов кольца R по *модулю* S. В следующем разделе мы увидим приложения теоремы B.2.

□ Циклические группы

До завершения подраздела остается еще одна тема для обсуждения. Пусть $(G, \cdot)$ — конечная группа, и пусть a — элемент из $G \setminus \{e\}$. Пусть $a^2, a^3, \ldots$ обозначают aa, aaa и т.д. Так как G конечна, существует единственное натуральное n, такое, что все элементы $e, a, a^2, \ldots, a^{n-1}$ различны, тогда как $a^n = a^j$ для некоторого $j, 0 \leq j < n$. Теперь мы покажем, что $j = 0$, т.е. что $a^n = e$. Допустим, что $j > 0$. Тогда из $a^n = a^j$ будет вытекать, что $a^{n-1} = a^{j-1}$. Это, однако, противоречит нашему определению числа n. Мы заключаем, что все элементы $a^i, 0 \leq i < n$, различны и что $a^n = e$.

Теперь очевидно, что элементы $e, a, a^2, \ldots, a^{n-1}$ образуют подгруппу H в G. Такая подгруппа H называется *циклической* подгруппой порядка n. Мы говорим, что элемент a порождает H и что он имеет (мультипликативный) порядок n.

Так как все элементы циклической группы — степени одного и того же элемента, циклическая группа коммутативна.

Лемма B.3
Пусть $(G, \cdot)$ — группа и a — элемент из G порядка n. Тогда для всех $m \in \mathbb{Z}$

$$a^m = e \Longleftrightarrow n | m.$$

Доказательство. Запишем $m = qn + r, 0 \leq r < n$. Тогда $a^m = e$, если и только если $a^r = e$, т.е. если и только если $r = 0$, т.е. если и только если $n | m$.

∎

Отсюда следует, что элемент a из G имеет порядок d тогда и только тогда, когда $a^d = e$ и $a^{d/p} \neq e$ для любого простого делителя p числа d.

Чтобы найти мультипликативный порядок целого числа a в $\mathbb{Z}_m^*$ (причем НОД$(a, m) = 1$), нужно — как вытекает из теоремы Эйлера (теорема A.14) и леммы B.3 — проверить лишь делители числа $\varphi(m)$. Следующий модуль делает это эффективным образом. Он использует функции *GCD*, *Divisors*, *EulerPhi* и *PowerMod* пакета "Mathematica".

```
MultiplicativeOrder[a_, m_] := If[GCD[a, m] == 1,
        Divisors[EulerPhi[m] ] //.
    {x_, y_} -> If[PowerMod[a, x, m] == 1, x, {y}]]
```

```
a = 2; m = 123456789;
n = MultiplicativeOrder[a, m]
```

```
6855006
```

Лемма B.4
Пусть $(G, \cdot)$ — группа и a элемент из G порядка n. Для $k > 0$ элемент a^k имеет порядок

$$\frac{n}{\text{НОД}(k, n)}.$$

Доказательство. Пусть m — порядок элемента a^k. Так как число $k/\text{НОД}(k, n)$ — целое, это влечет, что

$$(a^k)^{n/\text{НОД}(k,n)} = (a^n)^{k/\text{НОД}(k,n)} = e^{k/\text{НОД}(k,n)} = e.$$

Из леммы B.3 мы выводим, что m делит $n/\text{НОД}(k, n)$. Для доказательства обратного заметим, что $(a^k)^m = e$. Лемма B.3 влечет, что n делит km. Следовательно, $n/\text{НОД}(k, n)$ делит m.

■

Продолжая с теми же параметрами, что и выше, мы получаем

```
k = 3;
MultiplicativeOrder[a^k, m]
n / GCD[k, n]
```

```
2285002
```

```
2285002
```

Аналогично (B.1), для каждой подгруппы $(H, \cdot)$ конечной группы $(G, \cdot)$ можно определить отношение эквивалентности $\sim$ условием

$$a \sim b \quad \text{тогда и только тогда, когда} \quad ab^{-1} \in H.$$

Классы эквивалентности имеют вид

$$\{ha | h \in H\},$$

что можно легко проверить. Все они имеют ту же мощность, что и H. Отсюда следует, что число классов эквивалентности равно $|G|/|H|$. Вследствие этого $|H|$ делит $|G|$. Это доказывает следующую теорему.

Теорема B.5 (*Лагранж*)
Пусть $(G, \cdot)$ — конечная группа порядка n. Тогда порядок любой ее подгруппы $(H, \cdot)$ делит n. Кроме того, любой элемент $a \in G$ имеет порядок, делящий n.

B.1.2 Линейная алгебра

□ Векторные пространства и подпространства

Пусть $\mathbb{F}$ обозначает произвольное поле.

Определение B.6
Векторным пространством над $\mathbb{F}$ называется множество V объектов, которые можно складывать и умножать на элементы из $\mathbb{F}$ так, что результаты снова попадают в V. Кроме того, должны выполняться следующие свойства:

1. $(u + v) + w = u + (v + w)$ для всех $u, v, w \in V$;
2. В V существует нулевой элемент o, такой, что $v + o = o + v = v$ для всех $v \in V$;
3. Для каждого $v \in V$ существует такой элемент $-v \in V$, что $v + (-v) = (-v) + v = o$;
4. $u + v = v + u$ для всех $u, v \in V$;
5. $\alpha(u + v) = \alpha u + \alpha v$ для всех $u, v \in V$ и $\alpha \in \mathbb{F}$;
6. $(\alpha + \beta)v = \alpha v + \beta v$ для всех $\alpha, \beta \in \mathbb{F}$ и $v \in V$;
7. $(\alpha\beta)v = \alpha(\beta v)$ для всех $\alpha, \beta \in \mathbb{F}$ и $v \in V$;
8. $1 \cdot v = v$ для всех $v \in V$, где 1 обозначает единичный элемент поля $\mathbb{F}$.

Обычно элементы векторного пространства называют векторами, хотя они и не обязаны быть векторами в эвристическом смысле.

Примерами векторных пространств над $\mathbb{F}$ служат

i) множество $\mathbb{F}^n$ n-ок над $\mathbb{F}$;

ii) множество $\{f(x) \in \mathbb{F}[x] | \deg(f(x)) < n\}$ многочленов над $\mathbb{F}$ степени, меньшей n.

Часто из контекста бывает ясно, над каким полем определено данное векторное пространство. В этом случае поле в дальнейшем не упоминается.

Определение В.7
Подмножество W данного векторного пространства V называется *линейным подпространством* в V, если W само является векторным пространством относительно операций, уже определенных в V.

Чтобы определить, будет ли данное подмножество векторного пространства подпространством, не обязательно проверять все восемь свойств векторного пространства. Например, свойство 1 выполняется для всех $u, v, w \in W$, потому что оно заведомо выполняется для все элементов из V. Мы получаем теорему.

Теорема В.6
Подмножество W векторного пространства V является линейным подпространством тогда и только тогда, когда

i) $o \in W$;

ii) $u + v \in W$ для всех $u, v \in W$;

iii) $\alpha u \in W$ для всех $u \in W$ и $\alpha \in \mathbb{F}$.

Любое векторное пространство V имеет два тривиальных подпространства — $\{o\}$ и V.

Пусть V — векторное пространство и $v_1, v_2, \ldots, v_n$ — элементы из V. Выражение вида

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n, \qquad \text{где } \alpha_i \in \mathbb{F},$$

называется *линейной комбинацией* векторов $v_1, v_2, \ldots, v_n$.

Множество всех линейных комбинаций векторов $v_1, v_2, \ldots, v_n$ образует подпространство в V, о котором говорят, что оно *порождено* векторами $v_1, v_2, \ldots, v_n$ и обозначают через $< v_1, v_2, \ldots, v_n >$.

□ Линейная независимость, базисы и размерность

Наверное, наиболее важное понятие, связанное с векторными пространствами, — это понятие линейной (не)зависимости.

Определение B.8
Множество векторов $v_1, v_2, \ldots, v_n$ из векторного пространства V называют *линейно независимым*, если уравнение $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = o$ имеет лишь тривиальное решение $\alpha_1 = 0, \alpha_2 = 0, \ldots, \alpha_n = 0$. Множество векторов, не являющееся линейно независимым, называется *линейно зависимым*.

Допустим, что множество векторов $v_1, v_2, \ldots, v_n$ линейно зависимо. Тогда существует линейная комбинация $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = o$, где хотя бы одно $\alpha_i \neq 0$. Это позволяет нам записать

$$v_i = \alpha_i^{-1}(\alpha_1 v_1 + \ldots + \alpha_{i-1} v_{i-1} + \alpha_{i+1} v_{i+1} + \ldots + \alpha_n v_n).$$

Таким образом, мы получаем иное описание линейной зависимости.

Теорема B.7
Множество $v_1, v_2, \ldots, v_n$ векторов из векторного пространства V линейно зависимо тогда и только тогда, когда один из этих векторов можно выразить как линейную комбинацию других векторов этого множества.

В частности, это влечет, что любое множество векторов, включающее нулевой вектор o, линейно зависимо.

Теорема B.8
Предположим, что векторы $v_1, v_2, \ldots, v_n$ линейно независимы. Если заменить один из этих векторов его суммой с линейной комбинацией других векторов, то новое множество векторов вновь будет линейно независимо.

Пусть W — подпространство векторного пространства V и пусть $\{w_1, w_2, \ldots, w_n\} \subset W$.

Определение B.9
Множество $\{w_1, w_2, \ldots, w_n\}$ называется *базисом* для W, если

i) это множество векторов линейно независимо;

ii) $< w_1, \ldots, w_n >= W$, т.е. любой $w \in W$ является линейной комбинацией векторов $w_1, w_2, \ldots, w_n$.

В частности, если $W = V$, мы получаем базис для самого векторного пространства V. Например, в случае $V = \mathbb{F}^n$ следующее множество векторов образует базис для V:

$$e_1 = (1, 0, \ldots, 0), e_2 = (0, 1, 0, \ldots, 0), \ldots, e_n = (0, \ldots, 0, 1).$$

Этот базис обычно называют *стандартным базисом*.

В определении рассматриваются только конечные базисы. Но не каждое векторное пространство порождается конечным числом векторов.

Возьмем, например, $\mathbb{F} = \mathbb{R}$, а в качестве V — векторное пространство всех вещественно-значных функций на $\mathbb{R}$.

Можно доказать, что базис существует в любом векторном пространстве. Здесь мы коснемся только векторных пространств, порожденных конечным числом векторов. Следующая теорема весьма важна.

Теорема В.9
Допустим, что один базис подпространства W векторного пространства V содержит n векторов, а другой базис — m векторов. Тогда $n = m$.

Базис векторного пространства определен неоднозначно; однако число векторов в базисе определено однозначно.

Определение В.10
Если векторное пространство имеет базис из n векторов, мы называем n *размерностью* этого векторного пространства. Размерность нулевого векторного пространства $\{o\}$ по определению равна 0.

□ Скалярное произведение, ортогональность

Пусть V — векторное пространство над полем $\mathbb{F}$.

Определение В.11
Скалярным произведением на V называют билинейное отображение $V \times V \to \mathbb{F}$. Для векторов $u, v \in V$ образ обозначается через (u, v).

Билинейность означает, что для всех $u, v, w \in V$ и $\alpha \in \mathbb{F}$ должны выполняться следующие свойства:

$$\begin{aligned} (u+v, w) &= (u, w) + (v, w), \quad (u, v+w) = (u, v) + (u, w), \\ (\alpha u, v) &= \alpha(u, v) = (u, \alpha v). \end{aligned}$$

Это — очень общее определение скалярного произведения. В случаях $\mathbb{F} = \mathbb{R}$ или $\mathbb{F} = \mathbb{C}$ обычно требуют дополнительных свойств. Например, в вещественных векторных пространствах требуют положительной определенности для (u, u), т.е. чтобы для всех векторов $u \neq o$ было $(u, u) > 0$. В этом случае *длина* или *норма* вектора u определяется как $\sqrt{(u, u)}$ и часто обозначается через $||u||$.

Если $V = \mathbb{F}^n$, то стандартное скалярное произведение определяется равенством

$$(u, v) = u_1 v_1 + u_2 v_2 + \ldots + u_n v_n. \qquad (B.2)$$

Определение B.12

i) Два вектора u и v из V называются *ортогональными*, если $(u, v) = 0$.

ii) Два подпространства U и W из V называются *ортогональными*, если $(u, w) = 0$ для всех $u \in U$ и $w \in W$.

Если поле $\mathbb{F}$ конечно, то могут существовать такие ненулевые векторы u, что $(u, u) = 0$. Например, в векторном пространстве $\mathbb{F}^n$, где $\mathbb{F} = \{0, 1\}$, со стандартным скалярным произведением любой вектор с четным числом ненулевых координат ортогонален сам себе.

Пусть U — подпространство в V. Во многих приложениях полезно рассмотреть множество всех векторов, ортогональных к U.

Определение B.13

Ортогональное дополнение подпространства U из V, обозначаемое посредством $U^\perp$, — это множество всех векторов, которые ортогональны всем векторам из U.

Формула очевидна:

$$U^\perp = \{v \in V | (u, v) = 0 \quad \text{для всех } u \in U\}.$$

Для подпространств U и W конечномерного векторного пространства V выполняются следующие свойства:

Теорема B.10

i) Ортогональное дополнение к ортогональному дополнению подпространства совпадает с исходным подпространством, т.е. $(U^\perp)^\perp = U$.

ii) $\dim(U^\perp) = \dim(V) - \dim(U)$.

iii) Если $U \subset W$, то $W^\perp \subset U^\perp$.

iv) $(U \cap W)^\perp = U^\perp + W^\perp$.

В случае, когда $V = \mathbb{F}^n$ со стандартным скалярным произведением, мы имеем простое представление для $U^\perp$. Пусть $\{u_1, u_2, \ldots, u_m\}$ — базис в U и пусть A — $m \times n$-матрица со строками $u_1, \ldots, u_m$. Тогда мы имеем

$$v \in U^\perp \iff Av^T = o^T,$$

где верхний индекс T обозначает транспонирование вектора, т.е. получается вектор-столбец с теми же координатами, что и у v.

Определение В.14
Базис $\{v_1, v_2, \ldots, v_m\}$ векторного пространства V называется *ортогональным*, если все скалярные произведения (v_i, v_j), $i \neq j$, равны нулю.
Он называется *ортонормальным*, если дополнительно $||v_i|| = 1$ для всех $i, 1 \leq i \leq m$.

В.2 Конструкции

Множество целых чисел по модулю $m, m \in \mathbb{N} \setminus \{0\}$, которое было введено в разд. А.3, можно описать как кольцо классов вычетов $(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ (см. теорему В.2), поскольку $(m\mathbb{Z}, +, \cdot)$ — идеал коммутативного кольца $(\mathbb{Z}, +, \cdot)$. Кольцо классов вычетов коммутативно и имеет $< 1 >$ в качестве единичного элемента. Кольцо $(\mathbb{Z}/m\mathbb{Z}, +, \cdot)$ часто обозначается через $(\mathbb{Z}_m, +, \cdot)$.

Теорема В.11
Пусть m — натуральное число. Кольцо $(\mathbb{Z}_m, +, \cdot)$ является конечным полем (с m элементами) тогда и только тогда, когда m просто.

Доказательство. $\Longrightarrow$. Допустим, что m — составное, скажем, $m = ab, a > 1$ и $b > 1$. Тогда $< 0 >=< ab >=< a >< b >$, где $< a > \neq < 0 >$ и $< b > \neq < 0 >$. Поэтому кольцо $(\mathbb{Z}_m, +, \cdot)$ имеет делители нуля и, значит, не может быть полем.

$\Longleftarrow$. Допустим теперь, что m — простое (см. также пример В.3.). Мы должны доказать, что для любого класса эквивалентности $< a >$, $< a > \neq < 0 >$, существует такой класс эквивалентности $< b >$, что $< a >< b >=< 1 >$. Для этого достаточно показать, что для любого a при условии $m \not| a$ существует такой элемент b, что $ab \equiv 1 \pmod{m}$. Но это следует из леммы А.13 и теоремы А.18.

■

Удобства ради мы часто убираем скобки у представителей классов эквивалентности, так что a в действительности обозначает $< a >$.

Позже мы увидим, что для простого p кольцо $(\mathbb{Z}_p, +, \cdot)$ — в сущности единственное конечное поле из p элементов. Мы обозначаем его через $(\mathbb{F}_p, +, \cdot)$. В теории информации и связи часто работают с полем $\mathbb{F}_2$, состоящим из элементов 0 и 1.

Теперь мы продолжим строить конечные поля $\mathbb{F}_q$ для $q = p^m$, где p — простое.

Пусть $(F, +, \cdot)$ — коммутативное поле (не обязательно конечное) и $F[x]$ — множество многочленов над F, т.е. множество выражений

$$f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_n x^n,$$

где $f_i \in F, 0 \leq i \leq n$, и $n \in \mathbb{N}$. Наибольшее значение i, для которого $f_i \neq 0$, называется степенью многочлена $f(x)$.

Сложение и умножение многочленов определяется естественным образом:

$$\sum_i f_i x^i + \sum_i g_i x^i = \sum_i (f_i + g_i) x^i, \tag{B.3}$$

$$\left(\sum_i f_i x^i\right)\left(\sum_i g_i x^i\right) = \sum_k \left(\sum_{i+j=k} f_i g_j\right) x^k. \tag{B.4}$$

Пример B.4

Пусть $F = \mathbb{F}_2$ *и рассмотрим* $f(x) = 1 + x^2 + x^3$ *и* $g(x) = 1 + x + x^3$. *Тогда* $f(x) + g(x) = x + x^2$ *и* $f(x)g(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$.

В пакете "Mathematica" эти вычисления может выполнить функция PolynomialMod следующим образом:

```
p = 2; f = 1 + x^2 + x^3; g = 1 + x + x^3;
PolynomialMod[f + g, p]
PolynomialMod[f * g, p]
```

```
|| x + x^2
|| 1 + x + x^2 + x^3 + x^4 + x^5 + x^6
```

Следующая теорема проверяется непосредственно.

Теорема B.12
Пусть $(F, +, \cdot)$ — коммутативное поле. Тогда $(F[x], +, \cdot)$ — коммутативное кольцо с единичным элементом.

Понятия, определенные в приложении A для множества целых чисел, можно перенести на кольцо $(F[x], +, \cdot)$: делимость, приводимость (если многочлен может быть записан в виде произведения двух многочленов меньшей степени), неприводимость (что является аналогом простоты), НОД, теорема об однозначной факторизации (аналог основной теоремы теории чисел), алгоритм Эвклида, отношения сравнения и т.д. Детали мы оставляем читателю.

Здесь могут быть полезны следующие функции пакета "Mathematica": *PolynomialMod* (которая также приводит один многочлен по модулю другого), *Factor*, *PolynomialGCD*, *PolynomialLCM*. Их использование демонстрируется в следующих примерах.

```
p = 2; f = 1 + x + x^2 + x^7; g = 1 + x + x^3;
PolynomialMod[f, g, Modulus -> 2]
```

```
|| x + x^2
```

```
Factor[x^11 - 1, Modulus -> 3]
```

```
|| (2 + x) (2 + 2x + x^2 + 2x^3 + x^5) (2 + x^2 + 2x^3 + x^4 + x^5)
```

```
PolynomialGCD[1 + x^3, 1 + x^2, Modulus -> 2]
```

```
|| 1 + x
```

```
PolynomialLCM[1 + x^3, 1 + x^2, Modulus -> 2]
```

```
|| (1 + x^2) (1 + x + x^2)
```

Используя пакет *Algebra'PolynomialExtendedGCD'*, можно применить функцию *PolynomialExtendedGCD* пакета "Mathematica":

```
<< Algebra'PolynomialExtendedGCD'
```

```
PolynomialExtendedGCD[1 + x^3, 1 + x^2, Modulus -> 2]
```

```
|| {1 + x, {1, Mod[x, 2]}}
```

Некоторые важные выводы из теоремы В.12 формулируются в следующей теореме и ее следствии.

Теорема В.13
Пусть $a(x)$ и $b(x)$ — два многочлена из $F[x]$. Тогда в $F[x]$ существуют такие многочлены $u(x)$ и $v(x)$, что

$$u(x)a(x) + v(x)b(x) = \text{НОД}(a(x), b(x)).$$

Следствие В.14
Пусть $a(x)$ и $f(x)$ — два многочлена из $F[x]$, такие, что $\text{НОД}(a(x), f(x)) = 1$. Тогда сравнение

$$a(x)u(x) \equiv 1 \pmod{f(x)}$$

имеет единственное решение по модулю $f(x)$.

Решение указанного выше сравнения можно найти вновь с помощью функции *PolynomialExtendedGCD*. В самом деле, из

```
PolynomialExtendedGCD[1 + x^2, 1 + x + x^4, Modulus -> 2]
```

```
|| {1, {1 + x + x^3, x}}
```

можно заключить, что решением сравнения $(1 + x^2)u(x) \equiv 1 \pmod{1 + x + x^4}$ служит $1 + x + x^3$, что легко проверяется:

```
PolynomialMod[(1 + x^2) (1 + x + x^3), 1 + x + x^4,
Modulus -> 2]
```

```
|| 1
```

Еще одно важное свойство кольца $F[x]$ дает следующая теорема.

Теорема B.15
Любой многочлен степени $n > 0$ из $F[x]$ имеет не более n корней (или нулей) в F.

Доказательство. При $n = 1$ утверждение тривиально. Далее применяем индукцию по n.

Пусть $u \in F$ — корень многочлена $f(x)$ степени n над F (если такого u не существует, то ничего доказывать не надо). Запишем $f(x) = (x-u)q(x) + r(x)$, где $\deg(r(x)) < \deg(x-u) = 1$. Отсюда следует, что $r(x)$ — константа, скажем, r. Подстановка $x = u$ в равенство выше показывает, что $r = 0$. Поэтому $f(x) = (x-u)q(x)$.

Теперь $q(x)$ имеет степень $n-1$ и по индуктивной гипотезе $q(x)$ имеет не более $n-1$ корней в F. Поскольку поле не имеет делителей нуля, каждый корень многочлена $f(x)$ либо равен u, либо является корнем многочлена $q(x)$. Отсюда следует, что $f(x)$ имеет не более n корней в F. ■

Пусть $s(x)$ — ненулевой многочлен из $F[x]$. Легко проверить, что множество

$$\{a(x)s(x) | a(x) \in F[x]\}$$

образует идеал в кольце $(F[x], +, \cdot)$. Этот идеал мы обозначаем через $(s(x))$ и говорим, что он порождается многочленом $s(x)$.

Обратно, пусть $(S, +, \cdot)$ — произвольный идеал в $(F[x], +, \cdot)$, причем $S \neq F[x]$. Далее, пусть $s(x)$ — многочлен наименьшей степени из S. Возьмем любой другой многочлен $f(x)$ из S и запишем $f(x) = q(x)s(x) + r(x), \deg(r(x)) < \deg(s(x))$. Из свойств I (см. определение B.3) и R.1 (см. определение B.1) вытекает, что $r(x)$ — элемент из S. Из нашего предположения об $s(x)$ следует, что $r(x) = 0$ и поэтому $s(x)$ делит $f(x)$.

Из проведенного выше обсуждения следует, что любой идеал в кольце $(F[x], +, \cdot)$ порождается одним элементом. Кольцо с этим свойством называется *кольцом главных идеалов*.

Начиная с этого места, мы ограничимся конечными полями. До сих пор мы имели дело лишь с конечными полями $\mathbb{F}_p$ с простым p.

Пусть $f(x) \in \mathbb{F}_p[x]$ — многочлен степени n. Мы будем называть f *p-арным* многочленом. Пусть $(f(x))$ — идеал, порожденный многочленом $f(x)$. Благодаря теореме B.2, мы знаем, что $(\mathbb{F}_p[x]/(f(x)), +, \cdot)$ — коммутативное кольцо с единичным элементом $< 1 >$. Оно содержит p^n элементов, представленных p-арными многочленами степени $< n$.

Теорема B.16
Пустть $(\mathbb{F}_p, +, \cdot)$ — конечное поле из p элементов и $f(x)$ — многочлен степени n над $\mathbb{F}_p$. Тогда коммутативное кольцо $(\mathbb{F}_p[x]/(f(x)), +, \cdot)$ будет конечным полем (из p^n элементов) в том и только в том случае, когда $f(x)$ неприводим в $\mathbb{F}_p[x]$.

Доказательство. $\Longrightarrow$. Предположим, что $f(x) = a(x)b(x)$, где

$\deg(a(x)) > 0$ и $\deg(b(x)) > 0$. Тогда $< a(x) >< b(x) >=< a(x)b(x) >= =< f(x) >=< 0 >$, хотя $< a(x) >\neq< 0 >$ и $< b(x) >\neq< 0 >$. Таким образом, $(\mathbb{F}_p[x]/(f(x)), +, \cdot)$ — кольцо с делителями нуля. Следовательно, оно не может быть полем.

$\Longleftarrow$. С другой стороны, если $f(x)$ неприводим, любой ненулевой многочлен $a(x)$ степени $< n$ будет обладать мультипликативным обратным $u(x)$ по модулю $f(x)$ в силу следствия В.14. Для этого должно выполнятся $< a(x) >< u(x) >=< 1 >$. Отсюда следует, что $(\mathbb{F}_p[x]/(f(x)), +, \cdot)$ — поле. Мы уже знаем, что оно содержит p^n элементов.

■

Пример В.5

Пусть $q = 2$. *Поле* $\mathbb{F}_2$ *состоит из двух элементов* 0 *и* 1. *Пусть* $f(x) = 1 + x + x^3$. *Тогда* $(\mathbb{F}_2[x]/(1 + x + x^3), +, \cdot)$ — *конечное поле из* $2^3 = 8$ *элементов. Эти восемь элементов можно представить восемью бинарными многочленами степени* < 3. *Сложение и умножение выполняются по модулю* $1 + x + x^3$. *Например,*

$$(1+x+x^2)x^2 \equiv x^2+x^3+x^4 \equiv (1+x)(1+x+x^3)+1 \equiv 1 \pmod{1+x+x^3}.$$

Таким образом, x^2 — *мультипликативный обратный к* $1+x+x^2$ *в поле* $(\mathbb{F}_2[x]/(1 + x + x^3), +, \cdot)$.

В пакете "Mathematica" неприводимый многочлен над $\mathbb{F}_p$, p — *простое, можно найти с помощью функции* IrreduciblePolynomial, *для чего сначала нужно загрузить пакет* Algebra`FiniteFields`.

```
<< Algebra`FiniteFields`
```

```
p = 3; deg = 11;
IrreduciblePolynomial[x, p, deg]
```

|| $1 + x^9 + 2x^{10} + x^{11}$

В пакете "Mathematica" поле, определяемое многочленом $f(x)$ степени m, записывается в виде $\mathrm{GF}[p, \{f_0, f_1, \ldots, f_m\}]$. Сложение, вычитание, умножение и деление могут быть выполнены следующим образом:

```
f32 = GF[2, {1, 0, 1, 0, 0, 1}];
f32[{1, 0, 1, 0, 0}] + f32[{0, 1, 1, 0, 1}]
f32[{1, 0, 1, 0, 0}] - f32[{0, 1, 1, 0, 1}]
f32[{1, 0, 1, 0, 0}] * f32[{0, 1, 1, 0, 1}]
f32[{1, 0, 1, 0, 0}] / f32[{0, 1, 1, 0, 1}]
```

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

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

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

```
|| {1, 0, 1, 1, 0}_2
```

или следующим:

```
f32 = GF[2, {1, 0, 1, 0, 0, 1}];
x = f32[{0, 1, 0, 0, 0}];
x^5
x^6 + x^10
x^16* x^16
x^25 / x^22
```

```
|| {1, 0, 1, 0, 0}_2
|| {1, 1, 0, 1, 1}_2
|| {0, 1, 0, 0, 0}_2
|| {0, 0, 0, 1, 0}_2
```

В этот момент естественно возникают два вопроса:

1. Существует ли p-арный многочлен $f(x)$ степени n для любого простого числа p и любого натурального n? Если это так, то можно доказать существование конечных полей $\mathbb{F}_q$ для любой степени q простого числа.

2. Существуют ли другие конечные поля?

На первый вопрос утвердительный ответ будет дан в следующем разделе. Отрицательный ответ на второй вопрос будет дан в разд. B.4.

B.3 Число неприводимых многочленов над GF(q)

В этом разделе мы хотим подсчитать число неприводимых многочленов над конечным полем $\mathbb{F}_q$. Очевидно, если $f(x)$ неприводим, то таков же и $\alpha f(x)$ для $\alpha \in \mathbb{F}_q \setminus \{0\}$. Идеалы $(f(x)$ и $(\alpha f(x))$ в этом случае совпадают Поэтому мы будем подсчитывать только так называемые *нормированные* многочлены степени n, т.е. многочлены, у которых ведущий коэффициент (т.е. коэффициент при старшем члене x^n) равен 1.

Определение B.15
$I_q(n)$ = числу q-арных неприводимых нормированных многочленов степени n.
$I(n) = I_2(n)$ = числу бинарных неприводимых многочленов степени n.

Чтобы развить некоторую интуицию для нашей проблемы перечисления, начнем с грубой атаки в случае, когда $q = 2$. Мы попытаемся здесь определить $I(n)$.

Имеются только два бинарных многочлена степени 1, именно,

$$x \quad \text{и} \quad x+1.$$

По определению оба они неприводимы. Поэтому $I(1) = 2$.

Беря все возможные произведения x и $x+1$, находим три приводимых многочлена степени 2:

$$x \cdot x = x^2, \quad x \cdot (x+1) = x^2 + x \quad \text{и} \quad (x+1)^2 = x^2 + 1.$$

Так как имеется всего $2^2 = 4$ бинарных многочлена степени 2, отсюда следует, что существует только один неприводимый многочлен степени 2, именно, $x^2 + x + 1$. Поэтому $I(2) = 1$.

Каждый приводимый бинарный многочлен степени 3 может быть записан в виде произведения многочленов низших степеней $x, x+1$ и $x^2 + x + 1$. Таким путем получаются $x^i(x+1)^{3-i}, 0 \le i \le 3$, $(x^2 + x + 1)x$ и $(x^2 + x + 1)(x + 1)$. Так как имеются $2^3 = 8$ бинарных многочленов степени 3, мы заключаем, что среди них $8 - 4 - 2 = 2$ неприводимых бинарных многочлена степени 3. Поэтому $I(3) = 2$.

Два бинарных неприводимых многочлена степени 3 — следующие:

$$x^3 + x + 1 \qquad \text{и} \qquad x^3 + x^2 + 1.$$

В этот момент важно отметить, что приведенные выше перечислительные рассуждения не дают действительной формы неприводимых многочленов низших степеней. Мы можем только узнать, как много имеется многочленов данной степени.

В самом деле, можно подсчитать число $I(4)$ неприводимых многочленов 4-й степени следующим образом:

	число
• произведения четырех многочленов 1-й степени	5
• произведения одного многочлена 2-й степени и двух многочленов 1-й степени	3
• произведение двух многочленов 2-й степени	1
• произведения многочленов 3-й степени и 1-й степени	4
всего	= 13

Отсюда следует, что имеются $2^4 - 13 = 3$ неприводимых бинарных многочлена степени 4. Поэтому $I(4) = 3$.

Некоторыми дополнительными усилиями можно найти эти три неприводимых многочлена 4-й степени:

$$x^4 + x + 1, \qquad x^4 + x^3 + 1 \qquad \text{и} \qquad x^4 + x^3 + x^2 + x + 1.$$

Продолжая таким образом, при должном упорстве можно найти точные значения $I(5) = 6$, $I(6) = 9$ и т.д.

Приведенный выше метод не ведет к доказательству того, что $I(n) > 0$ для всех $n \in \mathbb{N}$ и не дает аппроксимации действительного значения $I(n)$. Начнем все заново.

Пусть $p_i(x), i = 1, 2, \ldots,$ — перечисление всех q-арных неприводимых нормированных многочленов, причем их степени образуют неубывающую последовательность. Следовательно, первые $I_q(1)$ имеют степень 1, следующие $I_q(2)$ многочленов имеют степень 2 и т.д.

Любой q-арный нормированный многочлен $f(x)$ имеет единственное разложение вида

$$\prod_{i=1}^{\infty}(p_i(x))^{e_i}, \quad e_i \in \mathbb{N}, i \geq 1,$$

где только конечное число e_i отлично от нуля. Отсюда следует, что $f(x)$ можно однозначно представить последовательностью $(e_1, e_2, \ldots)$. Пусть a_i будет степенью многочлена $p_i(x)$, а n — степенью многочлена $f(x)$. Тогда

$$e_1 a_1 + e_2 a_2 + \ldots = n.$$

Поэтому многочлены $f(x)$ находятся в однозначном соответствии с выражениями

$$(z^{a_1})^{e_1}(z^{a_2})^{e_2}\ldots$$

при разложении

$$(1 + z^{a_1} + z^{2a_1} + \ldots)(1 + z^{a_2} + z^{2a_2} + \ldots)\ldots,$$

т.е. при разложении

$$\prod_{i=1}^{\infty}(1 - z^{a_i})^{-1}.$$

Так как имеется в точности q^n q-арных нормированных многочленов степени n, сказанное выше доказывает равенство

$$\prod_{i=1}^{\infty}(1 - z^{a_i})^{-1} = 1 + qz + q^2z^2 + \ldots = (1 - qz)^{-1}$$

или, эквивалентно,

$$\prod_{i=1}^{\infty}(1 - z^{a_i}) = 1 - qz.$$

Введенное выше упорядочение показывает, что $a_i = k$ в точности для $I_q(k)$ значений i; поэтому соотношение выше можно переписать так:

$$\prod_{k=1}^{\infty}(1 - z^k)^{I_q(k)} = 1 - qz.$$

Возьмем теперь логарифмы от обеих частей и продифференцируем результат:

$$q(1 - qz)^{-1} = \sum_{k=1}^{\infty} kI_q(k)z^{k-1}(1 - z^k)^{-1}.$$

Умножение обеих частей на z дает

$$\sum_{n=1}^{\infty} q^n z^n = \sum_{k=1}^{\infty} kI_q(k) z^k (1 - z^k)^{-1} =$$

$$= \sum_{k=1}^{\infty} kI_q(k) \sum_{l=1}^{\infty} z^{kl} = \sum_{n=1}^{\infty} \sum_{k|n} kI_q(k) z^n.$$

Сравнение коэффициентов при степенях z в обеих частях дает соотношение

$$\sum_{k|n} kI_q(k) = q^n. \tag{B.5}$$

Теорема B.17

$$I_q(n) = \frac{1}{n} \sum_{d|n} \mu(d) q^{n/d}.$$

Доказательство. Примените к равенству B.5 формулу обращения Мёбиуса (теорема A.38). ■

Мы легко можем вычислить $I_q(n)$ в пакете "Mathematica" (используя *DivisorSum* и *MoebiusMu*).

```
DivisorSum[f_, n_] := Plus @@ (f / @ Divisors[n])
```

```
q = 2; m = 4; DSM[d_] := MoebiusMu[d] * q^(m/d);
(DivisorSum[DSM, m]) / m
```

```
3
```

Теперь довольно легко определить асимптотическое поведение функции $I_q(n)$ и доказать, что ее значения всегда положительны.

Прежде всего, $I_q(1) = q$, так как все нормированные многочлены степени 1 неприводимы по определению. Из соотношения (B.5) следует, что

$$q + nI_q(n) \leq \sum_{k|n} kI_q(k) = q^n.$$

Следовательно,

$$I_q(n) \leq \frac{q^n - q}{n}. \tag{B.6}$$

С другой стороны, соотношения (B.5) и (B.6) дают

$$q^n = \sum_{k|n} kI_q(k) \leq nI_q(n) + \sum_{k=0}^{\lfloor n/2 \rfloor} q^k < nI_q(n) + q^{1+n/2}.$$

Вместе с (B.6) это доказывает первое утверждение в следующей теореме.

Теорема B.18
Для всех n число $I_q(n)$ нормированных неприводимых многочленов степени n из $\mathbb{F}_q[x]$ удовлетворяет неравенствам

$$\frac{q^n}{n}(1-\frac{1}{q^{n/2-1}}) \leq I_q(n) \leq \frac{q^n}{n}(1-\frac{1}{q^{n-1}})$$

и

$$I_q(n) > 0.$$

Доказательство. То, что $I_q(n) > 0$ при $n \geq 3$, очевидно. При $n = 1$ и $n = 2$ это следует из теоремы B.17, но также прямо следует из условий $I_q(1) = q > 0$ и $I_q(2) = q^2 - \binom{q+1}{2} = \binom{q}{2} > 0$, что легко можно проверить непосредственно.

■

Следствие B.19

$$I_q(n) \approx \frac{q^n}{n}.$$

Читатель может проверить это приближение для отдельных случаев, используя пакет "Mathematica".

```
q = 2; m = 100; DSM[d_] := MoebiusMu[d] * q^(m/d);
N[(DivisorSum[DSM, m]) / q^m, 40]
```

```
0.9999999999999999111821579473501948675013
```

Отсюда следует, что случайно выбранный нормированный многочлен степени n неприводим с вероятностью около $1/n$. Функция *Factor* пакета "Mathematica" может легко проверить, будет ли конкретный многочлен неприводим.

```
Factor[1 + x + x^2 + x^3 + x^4, Modulus -> 2]
```

```
1 + x + x^2 + x^3 + x^4
```

B.4 Строение конечных полей

B.4.1 Циклическое строение конечного поля

Из теорем B.11, B.16 и B.18 следует, что конечные поля $(\mathbb{F}_q, +, \cdot)$ существуют для всех степеней q простых чисел. Если q — простое число, то $\mathbb{F}_q$ можно представить целыми числами по модулю q. Если q — степень простого числа, скажем, $q = p^m$, то $\mathbb{F}_q$ можно представить p-арными многочленами по модулю некоторого неприводимого многочлена степени m. Мы сформулируем это в виде теоремы.

Теорема B.20
Пусть p — простое число и $q = p^m, m \geq 1$. Тогда конечное поле порядка q существует.

Позже в этом разделе мы увидим, что любое конечное поле можно описать конструкцией из B.16. Но сначала мы докажем крайне важное свойство конечных полей, именно, что их мультипликативная группа циклична. В силу теоремы B.5 мы знаем, что любой ненулевой элемент в $\mathbb{F}_q$ имеет мультипликативный порядок, делящий $q-1$.

Определение B.16
Элемент ω конечного поля называется *n-м корнем* из единицы, если $\omega^n = 1$. Элемент ω называется *примитивным* n-м корнем из 1, если он имеет порядок n.
Если ω является примитивным $(q-1)$-м корнем из единицы, то ω называется *примитивным элементом* (или *образующим*) в $\mathbb{F}_q$.

Теорема B.21
Пусть $(\mathbb{F}_q, +, \cdot)$ — конечное поле и пусть d — целое число, делящее $q-1$. Тогда $\mathbb{F}_q$ содержит в точности $\varphi(d)$ элементов порядка d.
В частности, $(\mathbb{F}_q \setminus \{0\}, \cdot)$ — циклическая группа порядка $q-1$, которая содержит $\varphi(q-1)$ примитивных элементов.

Доказательство. По теореме B.5 любой ненулевой элемент в $\mathbb{F}_q$ имеет мультипликативный порядок d, который делит $q-1$. С другой стороны, предположим, что $\mathbb{F}_q$ содержит элемент порядка $d, d|(q-1)$, скажем, ω. Тогда все d различных степеней ω суть корни многочлена $x^d - e$. Из теоремы B.15 следует, что любой корень степени d из единицы в $\mathbb{F}_q$ является степенью ω. В предположении, что $\mathbb{F}_q$ содержит элемент порядка d, из леммы B.4 вытекает, что $\mathbb{F}_q$ содержит ровно $\varphi(d)$ элементов порядка d, именно, ω^i, где $\text{НОД}(i,d) = 1$.

Пусть $a(d)$ — число элементов порядка d в $\mathbb{F}_q$. Тогда сказанное выше означает, что

i) $a(d) = 0$ или $a(d) = \varphi(d)$,

а также что

ii) $\sum_{d|(q-1)} a(d) = q-1$.

С другой стороны, теорема A.12 утверждает, что $\sum_{d|(q-1)} \varphi(d) = q-1$. Поэтому заключаем, что $a(d) = \varphi(d)$ для всех $d|(q-1)$. В частности, $a(q-1) = \varphi(q-1)$, а это означает, что $\mathbb{F}_q$ содержит $\varphi(q-1)$ примитивных элементов и что $\mathbb{F}_q \setminus \{0\}$ — циклическая группа.

■

Чтобы убедиться, будет ли конкретный элемент ω из $\text{GF}(q)$ иметь порядок $d, d|(q-1)$, достаточно проверить, что $\omega^d = 1$ и что $\omega^{d/p} \neq 1$ для любого простого делителя p числа d. Посмотрите также обсуждение после леммы B.3.

Чтобы найти примитивный элемент в $\mathbb{Z}_p$ с простым p, можно использовать функцию *PowerList* пакета "Mathematica". Она находит примитивный элемент в $\mathbb{Z}_p$ и порождает все его степени (начиная с 0-й). Второй элемент в этом списке — сам примитивный элемент. Но сначала нужно загрузить пакет *Algebra'FiniteFields'*.

```
<< Algebra'FiniteFields'
```

```
p = 17; PrimeQ[p]
PowerList[GF[p, 1]][[2]]
```

```
True
```

```
{3}
```

Задачи B.6 и B.10 указывают эффективный путь (по Гауссу) нахождения примитивного элемента в конечном поле.

Следствие B.22
Любой элемент ω в $\mathbb{F}_q$ удовлетворяет равенствам

$$\omega^{q^n} = \omega, \qquad n \geq 1.$$

Доказательство. Для $\omega = 0$ утверждение тривиально истинно. В силу теорем B.5 и B.21 любой $\omega, \omega \neq 0$, имеет порядок, делящий $q-1$. Поэтому выполняется равенство $\omega^{q-1} = e$, а, значит, и $\omega^q = \omega$. Поскольку $\omega^{q^n} = (\omega^q)^{q^{n-1}}$, доказательство получается легким индуктивным рассуждением.

■

Следствие B.23
Пусть $\mathbb{F}_q$ — конечное поле. Тогда

$$x^q - x = \prod_{\omega \in \mathbb{F}_q} (x - \omega).$$

Доказательство. По следствию B.22 (посмотрите его доказательство) любой элемент ω из $\mathbb{F}_q$ является корнем многочлена $x^q - x$. Поэтому правая часть равенства выше делит его левую часть. Теперь равенство выполняется, потому что выражения в обеих сторонах нормированы и имеют одну и ту же степень.

■

Следствие B.23 будет использовано позже как инструмент для проверки того, будет ли некоторый элемент полей, содержащих $\mathbb{F}_q$, принадлежать в действительности самому $\mathbb{F}_q$.

Пример B.6

Рассмотрим конечное поле $(\mathbb{F}_2[x]/(f(x)), +, \cdot)$, *где* $f(x) = x^4 + x^3 + x^2 + x + 1$. *Оно содержит* $2^4 = 16$ *элементов, которые могут быть*

представлены бинарными многочленами степени < 4. *Элемент* x, *представляющий класс* $< x >$, *не является примитивным, так как* $x^5 \equiv \equiv (x+1)f(x)+1 \equiv 1 \pmod{f(x)}$. *Поэтому* x *имеет порядок* 5, *а не* 15. *В пакете "Mathematica" это можно проверить следующим образом:*

```
f = 1 + x + x^2 + x^3 + x^4;
PolynomialMod[x^2, f, Modulus -> 2]
PolynomialMod[x^3, f, Modulus -> 2]
PolynomialMod[x^4, f, Modulus -> 2]
PolynomialMod[x^5, f, Modulus -> 2]
```

```
|| x^2
|| x^3
|| 1 + x + x^2 + x^3
|| 1
```

Элемент $x+1$ *примитивен (его порядок равен* 15*), как можно увидеть в табл. В.1. Это также легко проверить. В самом деле,* $x+1$ *имеет порядок, делящий* 15. *Поэтому остается лишь проверить, что* $x+1$ *в степени* 3 *или* 5 *не приводится к* 1 *по модулю* $f(x)$.

```
f = 1 + x + x^2 + x^3 + x^4;
PolynomialMod[(x + 1)^3, f, Modulus -> 2]
PolynomialMod[(x + 1)^5, f, Modulus -> 2]
PolynomialMod[(x + 1)^15, f, Modulus -> 2]
```

```
|| 1 + x + x^2 + x^3
|| 1 + x^2 + x^3
|| 1
```

*Умножение легко выполняется с табл. В.*1. *Например,*

$$(1+x+x^2+x^3)(x+x^3) \equiv (x+1)^3(x+1)^{14} \equiv$$
$$\equiv (x+1)^{17} \equiv (x+1)^2 \equiv x^2+1 \pmod{f(x)}.$$

Элемент $x+1$ — *корень неприводимого многочлена* y^4+y^3+1, *так как*

$$(x+1)^4+(x+1)^3+1 \equiv 0 \pmod{f(x)}.$$

```
f = 1 + x + x^2 + x^3 + x^4;
PolynomialMod[(x + 1)^4 + (x + 1)^3 + 1, f, Modulus -> 2]
```

```
|| 0
```

Поэтому в $(\mathbb{F}_2/(g(x)), +, \cdot)$ *с* $g(x) = x^4 + x^3 + 1$ *элемент* x *примитивен; см. табл. B.2.*

	1	x	x^2	x^3
0	0	0	0	0
$(1+x)^0$	1	0	0	0
$(1+x)^1$	1	1	0	0
$(1+x)^2$	1	0	1	0
$(1+x)^3$	1	1	1	1
$(1+x)^4$	0	1	1	1
$(1+x)^5$	1	0	1	1
$(1+x)^6$	0	0	0	1
$(1+x)^7$	1	1	1	0
$(1+x)^8$	1	0	0	1
$(1+x)^9$	0	0	1	0
$(1+x)^{10}$	0	0	1	1
$(1+x)^{11}$	1	1	0	1
$(1+x)^{12}$	0	1	0	0
$(1+x)^{13}$	0	1	1	0
$(1+x)^{14}$	0	1	0	1

Таблица B.1. $(\mathbb{F}_2[x]/(1 + x + x^2 + x^3 + x^4), +, \cdot)$ с примитивным элементом $1 + x$.

	1	x	x^2	x^3
0	0	0	0	0
1	1	0	0	0
x	0	1	0	0
x^2	0	0	1	0
x^3	0	0	0	1
x^4	1	0	0	1
x^5	1	1	0	1
x^6	1	1	1	1
x^7	1	1	1	0
x^8	0	1	1	1
x^9	1	0	1	0
x^{10}	0	1	0	1
x^{11}	1	0	1	1
x^{12}	1	1	0	0
x^{13}	0	1	1	0
x^{14}	0	0	1	1

Таблица B.2. $(\mathbb{F}_2[x]/(1 + x^3 + x^4), +, \cdot)$ с примитивным элементом x.

В.4.2 Мощность конечного поля

Рассмотрим элементы $e, 2e, 3e$ и т.д. в $\mathbb{F}_q$. Так как $\mathbb{F}_q$ конечно, не все эти элементы различны. Также, если $ie = je$, где $i < j$, то $(j-i)e = 0$. Эти наблюдения оправдывают следующее определение.

Определение В.17
Характеристикой конечного поля $\mathbb{F}_q$ с единичным элементом e называется такое наименьшее натуральное число c, что $ce = 0$.

Теорема В.24
Характеристика конечного поля $\mathbb{F}_q$ — простое число.

Доказательство. Допустим, что характеристику c можно записать в виде $c'c''$, где $c' > 1$ и $c'' > 1$. Тогда $0 = ce = (c'e)(c''e)$, тогда как $c'e \neq 0$ и $c''e \neq 0$. Поэтому $c'e$ и $c''e$ — делители нуля. Это противоречит предположению, что $\mathbb{F}_q$ — поле.

■

Определение В.18
Два конечных поля $(\mathbb{F}_q, +, \times)$ и $(\mathbb{F}'_q, \oplus, \otimes)$ называются *изоморфными*, если существует взаимно однозначное отображение ψ из $\mathbb{F}_q$ на $\mathbb{F}'_q$ (поэтому $q = q'$), такое, что для всех ω_1 и ω_2 из $\mathbb{F}_q$

$$i) \qquad \psi(\omega_1 + \omega_2) = \psi(\omega_1) \oplus \psi(\omega_2),$$

$$ii) \qquad \psi(\omega_1 \times \omega_2) = \psi(\omega_1) \otimes \psi(\omega_2).$$

Говоря словами, два поля изоморфны, если после переименования их элементов они ведут себя совершенно одинаково по отношению к операциям сложения и умножения.

Лемма В.25
Пусть $(\mathbb{F}_q, +, \cdot)$ — конечное поле характеристики p. Тогда $(\mathbb{F}_q, +, \cdot)$ содержит подполе, изоморфное $(\mathbb{Z}_p, +, \cdot)$, т.е. множество целых чисел по модулю p.

Доказательство. Подмножество $\{ie | i = 0, 1, \ldots, p-1\}$ в $(\mathbb{F}_q, +, \cdot)$ образует подполе, которое изоморфно полю $(\mathbb{Z}_p, +, \cdot)$ относительно изоморфизма $\psi(ie) = i, 0 \leq i < p$.

■

Учитывая доказанную лемму, мы можем и будем отождествлять подполе в $(\mathbb{F}_q, +, \cdot)$ порядка p с полем $(\mathbb{Z}_p, +, \cdot)$. Подполе $\mathbb{F}_p$ часто называют *основным полем* для $\mathbb{F}_q$. Обратно, поле $\mathbb{F}_q$ называется *полем расширения* для $\mathbb{F}_p$.

Теорема В.26
Пусть $\mathbb{F}_q$ — конечное поле характеристики p. Тогда $\mathbb{F}_q$ — векторное пространство над $\mathbb{F}_p$ и $q = p^m$ для некоторого натурального $m \geq 1$.

Доказательство. Пусть $u_1, u_2, \ldots, u_m$ — базис в $\mathbb{F}_q$ над $\mathbb{F}_p$, т.е. любой элемент ω из $\mathbb{F}_q$ может быть записан в виде

$$\omega = a_1 u_1 + a_2 u_2 + \ldots + a_m u_m,$$

где $a_i \in \mathbb{F}_p, 1 \leq i \leq m$, и не существует линейной зависимости между элементами поля u_i над $\mathbb{F}_p$. Отсюда следует, что их представление однозначно и поэтому $q = |\mathbb{F}_q| = p^m$.

■

К этому времени мы знаем, что конечные поля $\mathbb{F}_q$ существуют только для степеней q простых чисел. Теорема B.20 утверждает, что $\mathbb{F}_q$ действительно существует для степени q простого числа. То, что все конечные поля с одним и тем же значением q изоморфны друг другу, будет доказано позже.

B.4.3 Некоторые правила вычислений над конечными полями; сопряженные элементы

Теорема B.27
Пусть ω — элемент из конечного поля $\mathbb{F}_q$ характеристики p. Тогда в $\mathbb{F}_q[x]$

$$(x - \omega)^p = x^p - \omega^p.$$

Доказательство. Пусть $0 < i < p$. Тогда НОД$(p, i!) = 1$, так что

$$\binom{p}{i} = \frac{p(p-1)\ldots(p-i+1)}{i!} \equiv 0 \pmod{p}$$

и, учитывая биномиальную теорему, мы получаем, что

$$(x - \omega)^p = x^p + (-\omega)^p = x^p - \omega^p,$$

где последнее равенство очевидно для нечетного p, а для $p = 2$ оно вытекает из равенства $+1 = -1$.

■

Чтобы продемонстрировать это, мы вновь используем функцию *PolynomialMod* пакета "Mathematica".

```
Clear[a, x];
p = 2; m = 3;
PolynomialMod[(x-a)^(p^m), p]
```

```
a^8 + x^8
```

Следствие В.28
Пусть $a_i, 1 \leq i \leq k$, — элементы конечного поля $\mathbb{F}_q$ характеристики p. Тогда для любого натурального n

$$\left(\sum_{i=1}^{k} a_i\right)^{p^n} = \sum_{i=1}^{k} a_i^{p^n}.$$

```
a = .; b = .; c = .;
p = 3; m = 3; PolynomialMod[(a + b + c)^(p^m), p]
```

```
a^27 + b^27 + c^27
```

Доказательство. Используйте индуктивное рассуждение по k и по n. Начните с равенства $(a_1 + a_2)^p = a_1^p + a_2^p$.

■

Следующая теорема дает мощный критерий для определения того, будет ли элемент из поля $\mathbb{F}_q$ характеристики p лежать в основном поле $\mathbb{F}_p$.

Теорема В.29
Пусть $\mathbb{F}_q$ — конечное поле характеристики p. Тогда $q = p^m, m > 0$, и $\mathbb{F}_q$ содержит $\mathbb{F}_p$ в качестве подполя. Пусть ω — элемент из $\mathbb{F}_q$. Тогда

$$\omega \in \mathbb{F}_p \Longleftrightarrow \omega^p = \omega.$$

Доказательство. По следствию В.23 p элементов подполя $\mathbb{F}_p$ удовлетворяют уравнению $x^p = x$. С другой стороны, многочлен $x^p - x$ по теореме В.15 имеет не более p корней в $\mathbb{F}_q$.

■

Пусть ω — элемент из $\mathbb{F}_q$, поля характеристики p, но ω не лежит в $\mathbb{F}_p$. Тогда $\omega^p \neq \omega$ в силу предыдущей теоремы. Но между ω^p и ω еще остается связь.

Теорема В.30
Пусть ω — элемент конечного поля $\mathbb{F}_q$ характеристики p. Пусть $f(x)$ — такой многочлен над $\mathbb{F}_p$, что $f(\omega) = 0$. Тогда для всех $n \in \mathbb{N}$

$$f(\omega^{p^n}) = 0.$$

Доказательство. Запишем $f(x) = \sum_{i=0}^{m} f_i x^i$. Так как $f_i \in \mathbb{F}_p, 0 \leq \leq i \leq m$, в силу следствия В.22 и теоремы В.29 получаем

$$0 = (f(\omega))^{p^n} = \left(\sum_{i=0}^{m} f_i \omega^i\right)^{p^n} = \sum_{i=0}^{m} (f_i \omega^i)^{p^n} =$$

$$= \sum_{i=0}^{m} f_i^{p^n} \omega^{ip^n} = \sum_{i=0}^{m} f_i (\omega^{p^n})^i = f(\omega^{p^n}).$$

В $\mathbb{R}$ и $\mathbb{C}$ имеется нечто подобное. Если $f(x)$ — многочлен с действительными коэффициентами и $f(\omega) = 0, \omega \in \mathbb{C}$, то также $f(\overline{\omega}) = 0$, где $\overline{\omega}$ — комплексно сопряженное к ω.

■

Следующая теорема утверждает, что число различных элементов ω^{p^i}, $i = 0, 1, \ldots$, зависит только от p и (мультипликативного) порядка элемента ω.

Теорема B.31
Пусть ω — элемент порядка n в конечном поле характеристики p. Пусть m — мультипликативный порядок числа p по модулю n, т.е. $p^m \equiv 1 \pmod{n}$, где $m > 0$. Тогда все m элементов

$$\omega, \omega^p, \omega^{p^2}, \ldots, \omega^{p^{m-1}}$$

различны и $\omega^{p^m} = \omega$.
Элементы $\omega^{p^i}, 0 \leq i \leq m-1$, называются *сопряженными* к ω.

Доказательство. Применяя лемму B.3 (дважды), получаем, что $\omega^{p^i} = \omega^{p^j}$ тогда и только тогда, когда $p^i \equiv p^j \pmod{n}$ и, значит, тогда и только тогда, когда $p^{i-j} \equiv 1 \pmod{n}$, т.е. тогда и только тогда, когда $i \equiv j \pmod{m}$.

■

Пример B.7

Рассмотрим $(\mathbb{F}_2[x]/(f(x)), +, \cdot)$, *где* $f(x) = x^4 + x^3 + x^2 + x + 1$ *(см. пример B.6). Элемент* x *поля имеет порядок* 5. *Мультипликативный порядок числа* 2 *по модулю* 5 *равен* 4. *Поэтому* x, x^2, x^{2^2} *и* x^{2^3} *различны, тогда как* $x^{2^4} = x$. *В самом деле,* $x^4 \equiv x^3 + x^2 + x + 1 \pmod{f(x)}$, $x^8 \equiv x^3 \pmod{f(x)}$ *в то время, как* $x^{16} \equiv x \pmod{f(x)}$, *как можно проверить с помощью функций* Table *и* PolynomialMod *пакета "Mathematica":*

```
p = 2; m = 4; f = 1 + x + x^2 + x^3 + x^4;
Table[PolynomialMod[x^p^i, f, Modulus -> p], {i, 0, m}] //
TableForm
```

```
x
x^2
1 + x + x^2 + x^3
x^3
x
```

В.4.4 Минимальные многочлены и примитивные многочлены

Теорема В.32
Пусть $\mathbb{F}_q$ — конечное поле характеристики p. Возьмем число $n|(q-1)$ и пусть ω — элемент порядка n из $\mathbb{F}_q$. Далее, пусть m — мультипликативный порядок числа p по модулю n. Тогда коэффициенты многочлена

$$m(x) = \prod_{i=0}^{m-1} (x - \omega^{p^i}) \qquad (B.7)$$

лежат в $\mathbb{F}_p$ и он неприводим над $\mathbb{F}_p$. Он называется *минимальным многочленом* элемента ω над $\mathbb{F}_p$.

Доказательство. Очевидно, $m(x)$ — многочлен над $\mathbb{F}_q$. Запишем $m(x) = \sum_{i=0}^{m} m_i x^i$. Мы должны показать, что коэффициенты m_i лежат в основном поле $\mathbb{F}_p$. С этой целью мы используем сильный критерий из теоремы В.29.

В силу теоремы В.27 и следствия В.22 (при $n = 1$) получаем

$$(m(x))^p = \prod_{i=0}^{m-1} \left(x - \omega^{p^i}\right)^p = \prod_{i=0}^{m-1} \left(x^p - \omega^{p^{i+1}}\right) =$$

$$= \prod_{i=1}^{m} \left(x^p - \omega^{p^i}\right) = \prod_{i=0}^{m-1} \left(x^p - \omega^{p^i}\right) = m(x^p).$$

Следовательно,

$$\sum_{i=0}^{m} m_i x^{pi} = m(x^p) = (m(x))^p = \left(\sum_{i=0}^{m} m_i x^i\right)^p = \sum_{i=0}^{m} m_i^p x^{pi}.$$

Сравнение коэффициентов при x^{pi} в обеих частях дает $m_i = m_i^p$. Из теоеремы В.29 следует, что $m_i \in \mathbb{F}_p, 0 \le i \le m$. Таким образом, $m(x)$ — многочлен из $\mathbb{F}_p[x]$.

Из теорем В.30 и В.31 следует, что никакой многочлен из $\mathbb{F}_p[x]$ степени, меньшей m, не может иметь ω в качестве корня. Поэтому $m(x)$ неприводим над $\mathbb{F}_p$.

■

Следствие В.33
Пусть ω — элемент порядка n в конечном поле характеристики p. Пусть $m(x)$ определен, как в теореме В.32, а $f(x)$ — многочлен с корнем ω. Тогда $f(x)$ делится на $m(x)$.

Доказательство. Скомбинируйте теоремы В.30, В.31 и В.32.

■

Таким образом, как определено в теореме В.32, $m(x)$ — нормированный многочлен наименьшей степени над $\mathbb{F}_p$, корнем которого служит ω. В этом — причина, почему $m(x)$ называется минимальным многочленом

для ω над $\mathbb{F}_p$. Его корнями являются ω и все сопряженные с ω элементы. Степень минимального многочлена $m(x)$ элемента ω часто называют просто степенью элемента ω над $\mathbb{F}_p$.

Если $m(x)$ — минимальный многочлен примитивного элемента, то $m(x)$ называют *примитивным многочленом*. "Mathematica" находит примитивный многочлен степени m над $\mathbb{F}_p$ от переменной z посредством функции *FieldIrreducible*.

```
<< Algebra`FiniteFields`
```

```
m = 6; p = 2;
FieldIrreducible[GF[p, m], z]
```

$$1 + z^5 + z^6$$

Пусть $f(x)$ — примитивный многочлен над $\mathbb{F}_p$ степени m. Таблица (подобная табл. В.2), в которой каждый ненулевой элемент конечного поля $(\mathbb{F}_p[x]/(f(x)), +, \cdot)$ представлен в виде многочлена от x степени $< m$, а также как степень x, называется таблицей логарифмов данного поля. Эти таблицы очень практичны, если в полях приходится выполнять обширные вычисления.

Таблицы логарифмов можно довольно легко сделать в пакете "Mathematica". В зависимости от того, находит ли сама "Mathematica" подходящий примитивный многочлен, или он задан, можно напечатать

```
p = 2;
TableForm[PowerList[GF[p, 4]]
```

```
1    0    0    0
0    1    0    0
0    0    1    0
0    0    0    1
1    0    0    1
1    1    0    1
1    1    1    1
1    1    1    0
0    1    1    1
1    0    1    0
0    1    0    1
1    0    1    1
1    1    0    0
0    1    1    0
0    0    1    1
```

или

```
p = 2;
TableForm[PowerList[GF[p, {1, 1, 0, 0, 1}]]]
```

```
1    0    0    0
0    1    0    0
0    0    1    0
0    0    0    1
1    1    0    0
0    1    1    0
0    0    1    1
1    1    0    1
1    0    1    0
0    1    0    1
1    1    1    0
0    1    1    1
1    1    1    1
1    0    1    1
1    0    0    1
```

Чтобы найти x^i в поле GF$[p, m]$, или, обратно, найти такое i, что x^i равно конкретному элементу в GF$[p, m]$, можно использовать функции *FieldExp*[GF$[p, m], i]$ и, соответственно, *FieldInd* (по существу для вычислений при значении *True* функции *PowerListQ*) пакета "Mathematica".

```
PowerListQ[GF[2, {1, 1, 0, 0, 1}]] == True;
f16 = GF[2, {1, 1, 0, 0, 1}];
FieldExp[f16, 5]
FieldInd[f16[{0, 1, 1, 0}]]
```

```
{0, 1, 1, 0}_2
5
```

Имеется несколько способов найти минимальный многочлен элемента поля. Мы продемонстрируем два метода.

Метод 1.

Пусть α — корень бинарного примитивного многочлена $x^5 + x^2 + 1$. Поэтому α имеет порядок 31, а сопряженными к α^3 служат α^6, α^{12}, α^{24} и α^{17}. Тогда минимальный многочлен элемента α^3 может быть найден программой

```
f := 1 + x^2 + x^5;
PolynomialMod[
(x - a^2)(x - a^6)(x - a^12)(x - a^24)(x - a^17), f, Modulus -> 2]
```

```
1 + x^2 + x^3 + x^4 + x^5
```

Метод 2.

Пусть α — корень бинарного примитивного многочлена $x^5 + x^2 + 1$. Чтобы найти минимальный многочлен элемента $\beta = \alpha^3$, мы сначала вычисляем $1, \beta, \beta^2, \beta^3, \beta^4$ и β^5, используя уравнение $\alpha^5 + \alpha^2 + 1 = 0$.

```
f = 1 + a^2 + a^5; b = a^3;
u0 = PolynomialMod[1, f, Modulus -> 2]
u1 = PolynomialMod[b, f, Modulus -> 2]
u2 = PolynomialMod[b^2, f, Modulus -> 2]
u3 = PolynomialMod[b^3, f, Modulus -> 2]
u4 = PolynomialMod[b^4, f, Modulus -> 2]
u5 = PolynomialMod[b^5, f, Modulus -> 2]
```

```
1
a^3
a + a^3
a + a^3 + a^4
a + a^2 + a^3
1 + a + a^2 + a^3 + a^4
```

Мы используем функцию *CoefficientList*, чтобы преобразовать коэффициенты в векторы. Заметим, что мы также используем функцию *Join*, чтобы дополнить векторы нулями, дабы сделать их все длиной 5.

```
M = {Join[[CoefficientList[u0, a], {0, 0, 0, 0}],
Join[CoefficientList[u1, a], {0}],
Join[CoefficientList[u2, a], {0}],
  CoefficientList[u3, a],
Join[CoefficientList[u4, a], {0}], CoefficientList[u5, a]};
MatrixForm[M]
```

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

Нам нужно найти линейную зависимость между $1, \beta, \beta^2, \beta^3, \beta^4$ и β^5, скажем, $\sum_{i=0}^{5} g_i \beta^i = 0$, где $g_i \in \mathrm{GF}(2)$. С этой целью мы используем функции *NullSpace* и *Transpose* пакета "Mathematica". Это приводит к минимальному многочлену $g(x)$ элемента β.

```
NullSpace[Transpose[M], Modulus -> 2]
```

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

Заключаем, что β имеет минимальный многочлен $1 + x^2 + x^3 + x^4 + +x^5$.

В.4.5 Дальнейшие свойства

Пусть $m(x)$ — минимальный многочлен элемента ω степени m. Из следствия В.33 вытекает, что p^m выражений $\sum_{i=0}^{m-1} f_i\omega^i, f_i \in \mathbb{F}_p, 0 \leq i \leq m$, принимают p^m различных значений. Для этих выражений сложение и умножение могут быть выполнены в точности, как в определениях (В.3) и (В.4), где нужно использовать соотношение $m(\omega) = 0$, чтобы привести степень результата к значению, меньшему m. Довольно легко проверить, что получается поле, изоморфное $(\mathbb{F}_q[x]/(m(x)), +, \cdot)$.

Если $m(x)$ примитивен, то получаем, что все элементы $1, x, \ldots, x^{(p^m-2)}$ различны по модулю $m(x)$, в точности как различны элементы $1, \omega, \ldots, \omega^{(p^m-2)}$. См. пример В.6, где примитивный элемент $\omega = 1 + x$ имеет минимальный многочлен $m(y) = 1 + y^3 + y^4$. Табл. В.2 представляет поле $(\mathbb{F}_q[x]/(m(x)), +, \cdot)$.

Лемма В.34
Пусть $m(x)$ — неприводимый многочлен степени m над полем с p элементами и пусть n кратно m. Тогда $m(x)$ делит $x^{p^n} - x$.

Доказательство. Рассмотрим кольцо классов вычетов $(\mathbb{F}_p[x]/(m(x)), +, \cdot)$. Это кольцо по теореме В.16 является полем с $q = p^m$ элементами. Элемент $< x >$ является корнем многочлена $m(x)$, так как $m(< x >) = =< m(x) >=< 0 >$. Из следствия В.22 (при $n = 1$) следует, что $< x >$ — корень многочлена $x^{p^n} - x$, $n \geq 1$. По следствию В.33 мы заключаем, что $m(x)$ делит $x^{p^n} - x$.

■

Справедливо также утверждение, обратное к лемме В.34.

Теорема В.35
Многочлен $x^{p^n} - x$ — это произведение всех неприводимых нормированных p-арных многочленов, степени которых делят n.

Доказательство. Пусть $m|n$. Существуют $I_p(m)$ неприводимых многочленов степени m над $\mathbb{F}_p$, все из которых делят $x^{p^n} - x$ по лемме В.34. Сумма их степеней равна $mI_p(m)$. Так как $\sum_{m|n} mI_p(m) = p^n = = \deg(x^{p^n} - x)$ в силу (В.5), неприводимые нормированные p-арные многочлены степеней m, $m|n$, образуют полное разложение многочлена $x^{p^n} - x$.

■

Пример B.8

Пусть $p = 2, n = 4$.
$I_2(1) = 2, I_2(2) = 1, I_2(4) = 3$ *(см. разд. B.3)*.
$x^{16} - x = x(x+1)(x^2+x+1)(x^4+x^3+x^2+x+1)(x^4+x^3+1)(x^4+x+1)$.

```
p = 2; m = 4;
Factor[x^p^m - x, Modulus -> p]
```

```
x(1 + x)(1 + x + x^2)(1 + x + x^4)(1 + x^3 + x^4)(1 + x + x^2 +
+ x^3 + x^4)
```

Следствие B.36
Пусть $f(x)$ — неприводимый многочлен в $\mathbb{F}_p[x]$ степени m и пусть $m|n$. Тогда конечное поле из p^n элементов содержит m корней многочлена $f(x)$.

Доказательство. По теореме B.35 $f(x)$ делит $x^q - x, q = p^n$. С другой стороны, в силу следствия B.23, $x^q - x = \prod_{\omega \in \mathbb{F}_q}(x - \omega)$.

■

Теорема B.37
Пусть p — простое число и $m \in \mathbb{N}$. Тогда конечное поле $\mathbb{F}_{p^m}$ единственно с точностью до изоморфизма.

Доказательство. Положим $q = p^m$ и пусть $\mathbb{F}_q$ — произвольное конечное поле порядка q. Пусть $f(x)$ — произвольный неприводимый p-арный многочлен степени m. Мы покажем, что $\mathbb{F}_q$ изоморфно полю $\mathbb{F}_p[x]/(f(x))$. По следствию B.36 $\mathbb{F}_q$ содержит m корней многочлена $f(x)$. Пусть ω — один из этих корней. Так как $f(x)$ неприводим в $\mathbb{F}_p[x]$, ни для какого многочлена над $\mathbb{F}_p$ меньшей степени ω не является корнем. Это влечет, что m элементов $1, \omega, \ldots, \omega^{m-1}$ независимы над $\mathbb{F}_p$. Поэтому любой элемент из $\mathbb{F}_q$ может быть записан в виде $\sum_{i=0}^{m-1} f_i\omega^i, f_i \in \mathbb{F}_p, 0 \leq i \leq m-1$.
Теперь изоморфизм между $\mathbb{F}_q$ и $\mathbb{F}_p[x]/(f(x))$ очевиден.

■

Следствие B.38
$\mathbb{F}_{p^m}$ изоморфно подполю в $\mathbb{F}_{p^n}$ тогда и только тогда, когда m делит n.

Доказательство. Все следующие утверждения эквивалентны:

i) $m|n$,

ii) $(p^m - 1)$ делит $(p^n - 1)$,

iii) $(x^{p^m} - x)$ делит $(x^{p^n} - x)$,

iv) $\prod_{\omega\in\mathbb{F}_{p^m}}(x-\omega)$ делит $\prod_{\omega\in\mathbb{F}_{p^n}}(x-\omega)$;

v) $\mathbb{F}_{p^m}$ — подполе поля $\mathbb{F}_{p^n}$.

■

Пример В.9

Из следствия В.38 вытекает, что $\mathbb{F}_{2^4}$ *содержит в качестве подполя* $\mathbb{F}_{2^2}$, *но не содержит* $\mathbb{F}_{2^3}$. *С помощью табл. В.2 можно легко проверить, что элементы* $0, 1, x^5, x^{10}$ *образуют подполе мощностью* 2^2 *в* $(\mathbb{F}_2[x]/(x^4+x^3+1), +, \cdot)$.

В.4.6 Многочлены деления круга

Рассмотрим конечное поле $\mathbb{F}_q$ характеристики p. Таким образом, $q = p^m$ для некоторого $m > 0$. По теореме В.5 любой элемент из $\mathbb{F}_q$ имеет порядок, делящий $q-1$. Пусть $n|(q-1)$ и ω — примитивный корень n-й степени из единицы в $\mathbb{F}_q$. Например, $\omega = \alpha^{(q-1)/n}$, где α — примитивный элемент в $\mathbb{F}_q$. Пусть $d|n$ и положим $\eta = \omega^{n/d}$. Тогда η — примитивный корень из единицы степени d. Очевидно, d элементов $1, \eta, \ldots, \eta^{d-1}$ — корни многочлена $x^d - 1$. По теореме В.15 никакой другой элемент из $\mathbb{F}_q$ не является корнем $x^d - 1$.

Определение В.19

Пусть $q = p^m, p$ — простое. Для любого $d|(q-1)$ p-арный *многочлен деления круга* (или циклотомический многочлен) $Q^{(d)}$ определяется равенством

$$Q^{(d)}(x) = \prod_{\xi\in GF(q)\text{порядка } d} (x-\xi).$$

Если ξ имеет порядок $d, d|(q-1)$, то по лемме В.4 ξ^p также имеет порядок d. Вместе с корнем ξ многочлена $Q^{(d)}(x)$ корнями являются и сопряженные с ξ элементы. Из теоремы В.32 следует, что $Q^{(d)}(x)$ является произведением некоторых минимальных многочленов над $\mathbb{F}_p$, и поэтому $Q^{(d)}(x)$ — многочлен над $\mathbb{F}_p$.

По теореме В.21 $Q^{(d)}(x)$ имеет степень $\varphi(d)$. Поскольку ω — примитивный корень n-й степени из единицы, получаем

$$x^n - 1 = \prod_{i=1}^{n-1}(x-\omega^i) = \prod_{\xi\in\mathbb{F}_q,\ \xi \text{ имеет порядок } n}(x-\xi) =$$

$$= \prod_{d|n}\prod_{\xi\in\mathbb{F}_q,\ \xi \text{ имеет порядок } d}(x-\xi) = \prod_{d|n} Q^{(d)}(x). \qquad (B.8)$$

Теорема В.39

$$Q^{(n)}(x) = \prod_{d|n}(x^d-1)^{\mu(n/d)}.$$

Доказательство. Нужно применить мультипликативную формулу обращения Мёбиуса (следствие А.39) к равенству (В.8).

■

Пример В.10

$$Q^{(36)}(x) = \prod_{d|36}(x^d - 1)^{\mu(36/d)} = \frac{(x^{36}-1)(x^6-1)}{(x^{18}-1)(x^{12}-1)} = \frac{x^{18}+1}{x^6+1} =$$

$$= x^{12} - x^6 + 1.$$

Это можно также вычислить в пакете "Mathematica":

```
DivisorProduct[f_, n_] := Times  @@ (f / @ Divisors[n])
```

```
n = 36; Clear[f, x];
f[d_] := (x^d-1)^MoebiusMu[n/d]
DivisorProduct[f, n] // Simplify
```

```
1 - x^6 + x^12
```

или прямо с функцией Cyclotomic из "Mathematica":

```
Cyclotomic[36, x]
```

```
1 - x^6 + x^12
```

Если $p = 2$, то можно записать $Q^{(36)}(x) = x^{12} + x^6 + 1$.

Выражение для $Q^{(n)}(x)$ в теореме В.39 кажется независимым от конечного поля. В действительности это не так, поскольку при вычислении значения этого выражения играет роль характеристика поля.

Все неприводимые делители многочлена $Q^{(d)}(x)$ имеют одну и ту же степень, так как все корни этого многочлена имеют один и тот же порядок d. В самом деле, по теореме В.32 каждый неприводимый делитель многочлена $Q^{(d)}(x)$ в качестве степени имеет мультипликативный порядок числа p по модулю d. В частности, мы имеем следующую теорему.

Теорема В.40
Число примитивных p-арных нормированных многочленов степени m равно

$$\frac{\varphi(p^m - 1)}{m}.$$

Доказательство. Примитивный p-арный многочлен степени m делит $Q^{(p^m-1)}(x)$, и этот многочлен деления круга имеет делители только данного типа. Степень многочлена $Q^{(p^m-1)}(x)$ равна $\varphi(p^m - 1)$.

■

Пример B.11

Пусть $p = 2$.

$$x^{16} - x = x(x^{15} - 1) = xQ^{(1)}(x)Q^{(3)}(x)Q^{(5)}(x)Q^{(15)}(x),$$

где

$$\begin{aligned} Q^{(1)}(x) &= x + 1, \\ Q^{(3)}(x) &= x^2 + x + 1, \\ Q^{(5)}(x) &= x^4 + x^3 + x^2 + x + 1, \\ Q^{(15)}(x) &= (x^4 + x + 1)(x^4 + x^3 + 1) \end{aligned}$$

В самом деле, имеются $\varphi(15)/4 = 2$ *примитивных многочлена степени* 4. *См. также пример* B.6.

Один из способов найти все примитивные многочлены степени m над $\mathbb{F}_p$ — факторизовать $Q^{(p^m-1)}(x)$.

Пример B.12

```
p = 2; m = 6; n = p^m - 1;
Factor[Cyclotomic[n, x], Modulus -> 2]
```

```
(1 + x + x^6)(1 + x + x^3 + x^4 + x^6)(1 + x^5 + x^6)
(1 + x + x^2 + x^5 + x^6)(1 + x^2 + x^3 + x^5 + x^6) (1 + x +
+ x^4 + x^5 + x^6)
```

Замечание. В этой главе мы рассмотрели $\mathbb{F}_q$, $q = p^m$ и p — простое, как расширение поля $\mathbb{F}_p$, однако все понятия, определенные в этой главе, можно обобщить на $\mathbb{F}_q[x]$. Таким образом, можно подсчитать число неприводимых многочленов степени n в $\mathbb{F}_q[x]$ или обсудить примитивные многочлены над $\mathbb{F}_q$ и т.д. Мы оставляем читателю проверку того, что все теоремы этого приложения можно обобщить: перенести с $\mathbb{F}_p$ и $\mathbb{F}_{p^m}$ на $\mathbb{F}_q$ и, соответственно, $\mathbb{F}_{q^m}$, просто заменяя p на q, а q — на q^m.

Пример B.13

Поле $\mathbb{F}_{16}$ *можно рассматривать как кольцо классов вычетов* $\mathbb{F}_4[x]/(x^2 + x + \alpha)$, *где* α — *элемент из* $\mathbb{F}_4$, *удовлетворяющий равенству* $\alpha^2 + \alpha + 1 = 0$.

B.5 Задачи

Задача B.1. Докажите, что $(\{x \in \mathbb{R} | x^2 \in \mathbb{Q}, x \neq 0\}, \cdot)$ — группа.

Задача B.2. Докажите, что элементы приведенной системы классов вычетов по модулю m образуют мультипликативную группу.

Задача B.3. Пусть $(G, *)$ — группа, а H — непустое подмножество в G. Тогда $(H, *)$ — подгруппа в $(G, *)$ тогда и только тогда, когда $h_1 * h_2^{-1} \in H$ для любых $h_1, h_2 \in H$.

Задача B.4. Докажите, что по существу имеются две различные группы порядка 4 (совет: каждый элемент имеет порядок, делящий 4).

Задача B.5. Найдите элемент порядка 12 в группе $(\mathbb{Z}_{13}^*, \times)$. Какие степени этого элемента имеют порядок 12? Ответьте на тот же вопрос для элементов порядков 6, 4, 3, 2 и 1.

Задача B.6. Пусть $(G, \cdot)$ — коммутативная группа. Пусть a и b — элементы из G порядков m и n соответственно.

a) Допустим, что $\text{НОД}(m, n) = 1$. Докажите, что $a \cdot b$ имеет порядок $m \times n$.

b) Допустим, что условие $\text{НОД}(m, n) = 1$ не выполняется. Определите целые числа s и t, такие, что $s|m, t|n$, $\text{НОД}(s, t) = 1$ и $\text{НОК}[s, t] = \text{НОК}[m, n]$.

c) Постройте элемент в G порядка $\text{НОК}[m, n]$.

Задача B.7^M. Найдите мультипликативный обратный к $1 + x^2 + x^3 \pmod{1 + x^2 + x^5}$ над GF(2) (совет: см. теорему B.13).

Задача B.8^M. Сколько имеется бинарных неприводимых многочленов, имеющих степени 7 и 8? (Совет: см. определение B.15 и теорему B.17).

Задача B.9. Создайте таблицу логарифмов поля $\text{GF}(2)[x]/(1 + x^2 + x^5)$ (совет: x — примитивный элемент). Используйте эту таблицу, чтобы выразить $x^{10} + x^{20}$ в виде степени x.

Задача B.10. Пусть $\alpha \in \text{GF}(q)$ имеет порядок $m, m < q - 1$. Какова вероятность того, что случайно взятый ненулевой элемент $\beta \in \text{GF}(q)$ имеет порядок n, делящий m? Дайте верхнюю оценку этой вероятности.

Постройте элемент порядка $\text{НОК}[m, n]$ (совет: см. задачу B.6) (фактически этот метод приводит к эффективному нахождению примитивного элемента в конечном поле; он принадлежит Гауссу).

Задача B.11. Какие подполя содержатся в GF(625)? Пусть α — примитивный элемент в GF(625). Какие степени α образуют различные подполя в GF(625)? (Совет: см. следствие B.38).

Задача B.12. Докажите, что над GF(2)

$$(x + y)^{2^k+1} = x^{2^k+1} + x^{2^k} \cdot y + x \cdot y^{2^k} + y^{2^k+1}.$$

(Совет: используйте следствие B.28).

Задача B.13. Сколько имеется бинарных примитивных многочленов степени 10? (Совет: см. теорему B.40).

Задача B.14. Найдите бинарный многочлен деления круга $Q^{(21)}(x)$ (совет: см. теорему B.39). Какова степень делителей многочлена $Q^{(21)}(x)$?

Задача B.15. Какова степень бинарного минимального многочлена примитивного корня из единицы степени 17? (Совет: теорема B.39). Сколько существует таких многочленов? Докажите, что каждый из них обратен самому себе. Найдите эти многочлены явным образом.

Задача B.16. Отображение *следа* Tr опреляется на $\mathrm{GF}(p^m)$, p — простое, формулой

$$\mathrm{Tr}(x) = x + x^p + x^{p^2} + \ldots + x^{p^{m-1}}.$$

a) Докажите, что $\mathrm{Tr}(x) \in \mathrm{GF}(p)$ для любого $x \in \mathrm{GF}(p^m)$ (совет: теорема B.29). Таким образом, Tr является отображением из $\mathrm{GF}(p^m)$ в $\mathrm{GF}(p)$.

b) Докажите, что отображение Tr линейно (совет: см. следствие B.28).

c) Докажите, что Tr принимает любое значение в $\mathrm{GF}(p)$ одинаково часто (совет: используйте теорему B.15).

d) Замените в этой задаче p на q, где q — степень простого числа, и проверьте те же самые утверждения.

Приложение C

Некоторые знаменитые математики

C.1 Эвклид из Александрии

Родился: около 365 г. до н.э. в Александрии, Египет.

Умер: около 300 г. до н.э.

Эвклид — наиболее выдающийся математик античности, известен, в основном, благодаря своему трактату по геометрии *"Начала"*. Многовековая устойчивость методического построения *"Начал"* делает Эвклида ведущим учителем математики всех времен.

О жизни Эвклида известно мало, за исключением того, что он работал в Александрии египетской. Приведенное выше изображение относится к 18-м веку, и его следует считать весьма условным.

Наиболее известная работа Эвклида — это его трактат по геометрии *"Начала"*. Это собрание знаний по геометрии, ставшее основой для обучения математике на 2000 лет. Скорее всего в *"Началах"* нет результатов, впервые доказанных самим Эвклидом, но организация материала и его представление несомненно принадлежат ему.

Тракат начинается с определений и аксиом. В их числе и знаменитая пятая аксиома параллельности, состоящая в том, что через данную точку можно провести ровно одну прямую, параллельную данной. Решение Эвклида сделать это утверждение аксиомой привело к созданию эвклидовой геометрии. Первая попытка отбросить пятый постулат была

предпринята только в 19-м веке, после чего началось изучение неэвклидовых геометрий.

Зенон из Сидона примерно через 250 лет после Эвклида писал: "Кажется, что в *"Началах"* впервые было показано, что предложения Эвклида не выводятся из одних только аксиом, и Эвклид сделал другие тонкие предположения."

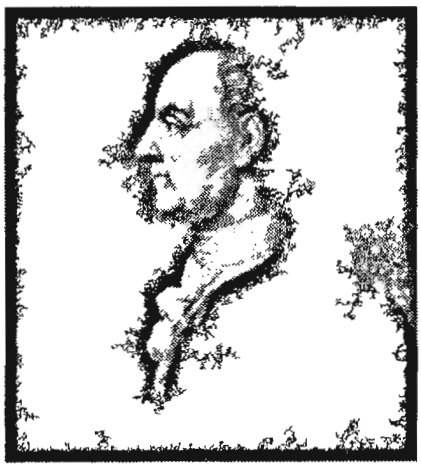
"Начала" разделены на 13 книг. Книги 1–6 посвящены геометрии на плоскости, книги 7–9 — теории чисел, книга 10 — теории иррациональных чисел, книги 11–13 — пространственной геометрии. Книга заканчивается обсуждением свойств пяти правильных многогранников и доказательством того, что их ровно пять. *"Начала"* Эвклида замечательны той ясностью, с которой формулируются и доказываются теоремы. Заданный Эвклидом стандарт строгости стал ориентиром для математиков более поздних веков.

С момента первой публикации *"Начал"* в 1482 г. было издано более тысячи редакций этой книги. Перу Эвклида принадлежат также дошедшие до нас *"Данные"* (содержащие 94 предложения), *"О частях"*, *"Оптика"* и *"Явления"*. Другие его книги: *"Местоположения поверхностей"*, *"Поризмы"*, *"Коники"*, *"Книга заблуждений"* и *"Начала музыки"* были утеряны.

Возможно Эвклид и не был первоклассным математиком, но многовековая устойчивость методического построения *"Начал"* делает Эвклида ведущим учителем математики всех времен.

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Euclid.html

С.2 Леонард Эйлер



Родился: 15 апреля 1707 г. в Базеле, Швейцария.

Умер: 18 сентября 1783 г. в Санкт-Петербурге, Россия.

Эйлер широко раздвинул границы современной аналитической геометрии и тригонометрии. Он внес решающий и формообразующий вклад в геометрию, математический анализ и теорию чисел.

Отец Эйлера был священником и хотел, чтобы сын пошел по его стопам. Для подготовки к служению Эйлера направили учиться в Базельский университет. Однако вскоре его любимым предметом стала геометрия. Эйлер с помощью Иоганна Бернулли получил согласие своего отца на занятие математикой. Иоганн Бернулли стал его учителем.

Эйлер был приглашен в Санкт-Петербургскую академию наук в 1727 г., т.е. через два года после ее основания Екатериной I — женой Петра Великого. С 1727 по 1730 гг. он служил лейтенантом медицины в российском флоте. В Санкт-Петербурге он жил вместе с Даниэлем Бернулли. В 1730 г. Эйлер стал профессором физики в академии наук, а в 1733 г. — профессором математики. В том же году он женился и покинул дом Даниэля Бернулли. У него было 13 детей, из которых 8 умерли в раннем детстве. Он утверждал, что сделал некоторые из своих великих открытий, когда держал младенца на руках, а другие дети играли у его ног.

Свою основную математическую деятельность Эйлер начал с публикации множества статей и книги *"Механика"* (1736–73 гг.), в которой впервые динамика Ньютона подробно представлена в форме математического анализа.

В 1741 г. по приглашению Фридриха Великого Эйлер перешел в Берлинскую академию наук, где проработал 25 лет. Даже живя в Берлине, он получал часть своего жалования из России и никогда не ладил с Фридрихом. За период работы в Берлине он написал свыше 200 статей, три книги по математическому анализу и научно-популярную книгу *"Письма ... к некоей немецкой принцессе"* в трех томах (1768–72 гг).

В 1766 г. Эйлер вернулся в Россию. Он спорил с Фридрихом Великим по поводу академической свободы, и Фридрих был сильно разгневан из-за его отъезда. В возрасте 31 года у Эйлера перестал видеть правый глаз, а после возвращения в Санкт-Петебург Эйлер практически полностью ослеп после операции на катаракте. Благодаря своей выдающейся памяти, он смог продолжить исследования по оптике, алгебре и движению Луны. Удивительно, что будучи практически слепым, он после 1765 г. (когда ему было 58) выпустил почти половину всех своих работ.

После смерти Эйлера в 1783 г. академия наук Санкт-Петербурга еще в течение 50 лет издавала его ранее неопубликованные работы.

Эйлер широко раздвинул границы современной аналитической геометрии и тригонометрии. Он внес решающий и формообразующий вклад в геометрию, математический анализ и теорию чисел. Большая работа в области теории чисел проделана им в переписке с Гольдбахом. Эйлер объединил дифференциальное исчисление Лейбница и метод флюксий (производных) Ньютона в математический анализ. В теории чисел он доказал теорему о простых числах и открыл закон биквадратичной взаимности.

Он был наиболее плодовитым математиком всех времен. Его труды составили 886 книг и статей.

Именно им введены обозначения $f(x)$ для функции (1734 г.), e для основания натурального логарифма (1727 г.), i для квадратного корня из -1 (1777 г.), π для числа "пи", $\sum$ для суммы (1755 г.), и т.д. Он также ввел бета- и гамма-функции, интегрирующий множитель для дифференциальных уравнений и многое другое.

Он изучал континуальную механику, теорию движения Луны вместе с Клеро, задачу о движении трех тел, теорию упругости, акустику, волновую теорию света, гидравлику, музыку и т.д. Он заложил основы аналитической механики в своей *"Теории движения твердых тел"* (1765 г.).

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Euler.html

С.3 Пьер де Ферма

Родился: 17 августа 1601 г. в Бомон-де-Ломань, Франция.

Умер: 18 января 1665 г. в Кастре, Франция.

Отец **Пьера Ферма** был богатым торговцем кожей и вторым консулом в Бомон-де-Ломань. У Пьера были брат и две сестры. Он рос в родном городе. Если и есть какие-либо свидетельства его школьного образования, то они должны быть в местном францисканском монастыре. Он учился в университете Тулузы до переезда в Бордо во второй половине 1620-х гг. В Бордо он начал свои первые математические исследования и в 1629 г. представил одному из местных математиков копию реставрированной работы Аполлония *"Plane Loci"*. В Бордо он общался с Бограном. В этод период Ферма выполнил важную работу о максимумах и минимумах, которую затем представил Этьену д'Эспанье, разделявшему его математические интересы.

Из Бордо Ферма переехал в Орлеан, где изучал юриспруденцию в университете. Он получил степень гражданского юриста и купил место советника парламента Тулузы. Таким образом, с 1631 г. Ферма стал юристом и правительственным чиновником и получил право изменить имя с Пьер Ферма на **Пьер де Ферма**.

Остаток своей жизни он прожил в Тулузе. Он работал в Тулузе, в своем родном городе Бомон-де-Ломань и соседнем городе Кастре. С 14 мая 1631 г., момента своего назначения, Ферма работал в нижней палате парламента, 16 января 1638 г. он был назначен в верхнюю палату, а, затем, в 1652 г., выдвинулся на высший уровень в криминальном суде. Дальнейшее быстрое продвижение по службе, по-видимому, объясняется его возрастом и эпидемией чумы, поразившей регион в начале 1650-х гг. и приведшей к гибели множества стариков. Ферма и сам тоже заразился, и в 1653 г. появилось ошибочное сообщение о его смерти. Затем оно было исправлено:

Ранее я сообщил тебе о смерти Ферма. Он жив, и мы больше не беспокоимся за его здоровье, хотя еще недавно мы считали его мертвым.

Следующее сообщение, отправленное Кольберу, значимой фигуре во Франции того времени, абсолютно справедливо:

Ферма — человек большой эрудиции, общающийся с широко образованными людьми. Но он сильно чем-то поглощен, не сообщает достаточных подробностей и ошибается.

Разумеется, Ферма был поглощен математикой. Он поддерживал математическое общение с Бограном и после переезда в Тулузу, кроме того, там он нашел нового интересующегося математикой друга — Каркави. Они познакомились на работе. Каркави, как и Ферма, был муниципальным чиновником в Тулузе, и разделял его любовь к математике. Ферма рассказывал Каркави о своих математических открытиях.

В 1636 г. Каркави переехал в Париж в качестве королевского библиотекаря и познакомился с Мерсенном и его группой. Каркави вызвал интерес Мерсенна описаниями открытий Ферма в области падения тел, и Мерсенн написал Ферма. Ферма откликнулся 26 апреля 1636 г. и, вдобавок, сообщил Мерсенну об ошибках, которые, как ему казалось, сделал Галилей при описании свободного падения. Кроме того, он сообщил Мерсенну о своей работе, касающейся спиралей, и о реставрации работы Аполлония *"Plane Loci"*. Его работа о спиралях была мотивирована рассмотрением траекторий свободно падающих тел. В ней он для вычисления площади под спиралью обобщил и применил метод из работы Архимеда *"О спиралях"*. В дополнение Ферма написал:

Я также обнаружил множество способов решения различных теоретико-числовых и геометрических задач, для которых не достаточно техники Виета. Я поделюсь этим с вами, как только вы этого пожелаете, причем сделаю это безо всякого честолюбия, от которого я более свободен и более далек, чем любой другой человек на свете.

Есть некая ирония судьбы в том, что этот первоначальный контакт Ферма с научным сообществом произошел на почве исследований свобод-

ного падения, так как Ферма проявлял весьма незначительный интерес к физическим приложениям математики. Даже излагая свои результаты о свободно падающих телах, он больше интересовался доказательством геометрических теорем, нежели их связью с реальным миром. Кроме того, это первое письмо содержало две задачи на максимумы, которые Ферма попросил Мерсенна передать парижским математикам. Это стало обычным стилем писем Ферма: он предлагал другим получить те результаты, которые он сам уже получил.

Роберваль и Мерсенн обнаружили, что задачи Ферма в первом и в последующих письмах крайне сложны и обычно не решаются с использованием известных способов. Они попросили его раскрыть свои методы, и Ферма переслал парижским математикам *"Метод определения максимумов и минимумов и построения касательных к кривым"*, реставрированную работу Аполлония *"Plane Loci"* и свой алгебраический подход к геометрии, содержащийся в работах *"Введение в плоскость"* и *"Solid Loci"*.

У него быстро появилась репутация одного из ведущих математиков мира, но попытки издать его работы оказывались в основном неудачными из-за того, что он сам никогда не имел особого желания их публиковать. Тем не менее некоторые из его методов были опубликованы. Так, например, Эригон к своему главному труду *"Математика круга"* добавил приложение, содержащее метод Ферма для определения максимумов и минимумов. Переписка между Ферма и другими математиками вызывала не только похвалы в его адрес. Франк де Бесси был раздражен задачами Ферма, казавшимися ему неразрешимыми. Рассерженный, он написал Ферма, и хотя Ферма в своем ответе сообщил некоторые дополнительные детали, Франк де Бесси почувствовал, что Ферма над ним почти издевается.

Однако, вскоре Ферма начал вступать в спор с более заметными математиками, чем Франк де Бесси. Получив от Бограна копию работы Декарта *"Диоптрика"*, Ферма не обратил на нее заметного внимания, поскольку был занят перепиской с Робервалем и Этьенном Паскалем о методах интегрирования и их применении при нахождении центра тяжести. Мерсенн попросил его высказать свое мнение по поводу "Диоптрики", и Ферма охарактеризовал эту работу как

блуждание наощупь в темноте.

Он объявил, что Декарт неверно вывел законы преломления из-за того, что опирался на неверные предположения. Можно сказать, что Декарту не слишком понравилась такая оценка. Вскоре у Декарта появился повод для еще большей обиды, поскольку он обнаружил, что работа Ферма о максимумах, минимумах и построении касательных принизила важность его собственной работы *"Геометрия"*, которой Декарт больше всего гордился и которой пытался показать, что может дать его *"Рассуждение о методе"*.

Декарт набросился на метод Ферма определения максимумов и минимумов и нахождения касательных. В спор оказались вовлечены Роберваль и Этьенн Паскаль, а затем и Дезарг, которого Декарт попросил быть арбитром. Ферма привел корректное доказательство и, наконец, Декарт признал его правоту, написав:

... рассмотрев последний метод, который вы используете для построения касательных к кривым, я не могу сказать ничего другого кроме того, что он очень хорош, и, что если бы вы с самого начала обосновывали его так, я бы вообще не стал с вами спорить.

Может на этом дело кончилось, и положение Ферма укрепилось? Отнюдь, поскольку Декарт по-прежнему пытался навредить репутации Ферма. Например, хотя он, написав Ферма, похвалил его работу по определению касательной к циклоиде (которая и на самом деле верная), написав Мерсенну, Декарт объявил эту работу ошибочной, и сказал, что Ферма неадекватен как математик и как мыслитель. Декарт был весьма уважаем, поэтому он мог весьма существенно навредить репутации Ферма.

В период с 1643 по 1654 гг. Ферма прервал контакт со своими парижскими научными коллегами. Для этого было несколько причин. Во-первых, нагрузка на работе не позволяла ему посвящать много времени математике. Во-вторых, с 1648 г. в Тулузе стало сказываться заметное влияние фронды и гражданской войны во Франции. И, наконец, чума 1651 г. привела к тяжелым последствиям как для Тулузы, так и для самого Ферма. Тем не менее в течение всего этого времени Ферма продолжал исследования в теории чисел.

Ферма наиболее известен своими работами в теории чисел, в частности, благодаря последней (большой) теореме Ферма. Эта теорема утверждает, что уравнение $x^n + y^n = z^n$ относительно неизвестных x, y и z не имеет ненулевых целочисленных решений при $n > 2$. Ферма написал на полях *"Арифметики"* Диофанта в переводе Баше:

Я нашел воистину замечательное доказательство, но эти поля слишком малы, чтобы его вместить.

Это замечание на полях стало известно после того, как сын Ферма Самюэль в 1670 г. опубликовал перевод Баше *"Арифметики"* Диофанта с замечаниями отца.

Теперь считается, что "доказательство" Ферма было ошибочным, хотя невозможно утверждать этого наверняка. Истинность утверждения Ферма была доказана в июне 1993 г. британским математиком Эндрю Уайлсом, но в конце 1993 г., когда появились проблемы, Уайлс объявил, что у него нет доказательства. В ноябре 1994 г. Уайлс вновь объявил, что нашел верное доказательство, которое теперь общепризнано.

Неудачные попытки доказательства теоремы в течение более, чем 300 лет, привели к открытию теории коммутативных колец и изобилию других математических открытий.

Переписка Ферма с парижскими математиками возобновилась в 1654 г., когда Блэз Паскаль, сын Этьенна Паскаля, написал ему, попросив подтвердить его мысли о вероятности. Блэз Паскаль знал о Ферма от своего отца, который умер за три года до этого, и был прекрасно осведомлен о выдающихся математических способностях Ферма. В их короткой переписке были заложены основы теории вероятностей, поэтому их двоих теперь считают основателями этой теории. Ферма же, чувствуя свою изоляцию и желая вновь вернуться к своему привычному стилю общения с математиками, с вызовом попытался изменить тему с вероятности на теорию чисел. Паскаль не заинтересовался, но Ферма, не понимая этого, написал Каркави следующее:

Я рад возможности ознакомиться с мнениями, согласующимися с мнением г. Паскаля, так как я бесконечно уважаю его гений... вы двое можете быть уверены, что в публикации, в которой я согласился с вашим руководством, вы можете прояснять и дополнять все, что бы ни показалось вам слишком кратким, этим вы снимете с меня тот труд, за который мне не дают взяться мои служебные обязанности.

Однако Паскаль совершенно не собирался редактировать работу Ферма, и после этого короткого сообщения о желании опубликовать свою работу Ферма вновь отказался от этой идеи. После этого он пошел еще дальше в постановке своих задач:

Две математические задачи, считающиеся французскими, английскими, голландскими и другими европейскими математиками нерешаемыми, поставлены господином де Ферма, королевским советником в парламенте Тулузы.

Его задачи не вызвали слишком большого интереса, поскольку большинство математиков не считали теорию чисел важной темой. Тем не менее вторая из задач, состоящая в том, чтобы найти все целые решения уравнения $N \cdot x^2 + 1 = y^2$ для N, не являющегося точным квадратом, была решена Уоллисом и Браункером. В процессе решения они развили теорию цепных дробей. Браункер дал рациональные решения, которые привели к спорам. Вероятно, Фрэнк де Бесси был единственным математиком того времени, проявившим реальный интерес к теории чисел, но он не обладал математическими талантами, достаточными для того, чтобы внести значительный вклад в эту теорию.

Ферма продолжал ставить задачи, а именно, что сумма двух кубов не может быть кубом (это частный случай большой теоремы Ферма, который может указывать, что в то время Ферма понял, что его доказательство общего результата содержит ошибку), что сущетсвует ровно два целых решения уравнения $x^2 + 4 = y$, и что уравнение $x^2 + 2 = y^3$ имеет ровно одно целочисленное решение. Он ставил задачи прямо на английском. Никто не замечал, что Ферма надеялся, что его узко сформулированные задачи могут привести к открытию более глубоких теоретических результатов.

В это время один из студентов Декарта, собиравший его письма для издания, обратился к Ферма с просьбой ознакомиться с перепиской между Ферма и Декартом. Это заставило Ферма вновь вспомнить свои аргументы, использованные 20 лет назад, и свои возражения к оптике Декарта. В частности, он был недоволен тем, как Декарт описывал преломление света. Теперь же он был согласен с законом синусов для преломления светового луча, предложенным Снеллом и Декартом. Однако Ферма теперь и сам вывел этот закон из более общего свойства света, а именно, из того, что свет всегда распространяется по кратчайшему пути. В то время этот принцип Ферма, теперь ставший одним из основополагающих оптических свойств, не нашел одобрения у математиков.

В 1656 г. Ферма начал переписываться с Гюйгенсом. Это произошло из-за интереса Гюйгенса к вероятности, но вскоре стараниями Ферма в переписке возникла тема теории чисел. Эта тема не интересовала Гюйгенса, но Ферма очень старался его увлечь, и в посланном в 1659 г. Гюйгенсу через Каркави *"Новом отчете об открытиях в науке о числах"* он раскрыл существенно больше своих методов, чем обычно.

Ферма описал свой метод бесконечного спуска и представил пример его использования в доказательстве того, что каждое число вида $4k+1$ может быть записано в виде суммы двух квадратов. Допустив, что некоторое число вида $4k+1$ не может быть записано в виде суммы двух квадратов, он утверждал, что найдется меньшее число вида $4k+1$, которое тоже нельзя представить в виде суммы двух квадратов. Последовательное использование этого аргумента ведет к противоречию. В этом письме Ферма не объяснил, как меньшее число строится из большего. Предполагают, что Ферма знал, как сделать этот шаг, но вновь его нежелание представить свой метод вызвало потерю интереса к этому со стороны математиков. Интерес не появлялся до тех пор, пока Эйлер не взялся за эти задачи и не заполнил недостающие шаги.

Ферма описывают так:

*Скрытный и молчаливый, он не любит говорить о себе и ненавидит раскрывать свои мысли. ... Его мысли, хотя оригинальные и новые, находились в пределах возможностей, ограниченных тем временем [*1600–1650 *гг.] и тем местом [Франция].*

Карл Б. Бойер говорил:

Признание значения работы Ферма в анализе запоздало, в частности, потому, что он придерживался системы обозначений, изобретенной Франсуа Виетом, а они устарели после выхода "Геометрии" Декарта. Помехи, вызванные неудобными обозначениями, менее жестко сказывались в любимой области исследований Ферма — теории чисел, но здесь он не нашел того, кто бы разделил его энтузиазм.

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Ferma.html

С.4 Эварист Галуа

Родился: 26 октября 1811 г. в Бур-ля-Рен (около Парижа), Франция.
Умер: 31 мая[1] 1832 г. в Париже, Франция.

Известный своим вкладом в теорию групп, **Эварист Галуа** разработал метод определения, когда уравнение общего вида разрешимо в радикалах.

Отец Галуа, Никола Габриэль Галуа, и его мать, Аделаида Мари Демантe, оба были людьми интеллигентными и прекрасно образованными в области философии, классической литературы и религии. Однако ни у кого из членов семьи Галуа не было отмечено каких-либо математических способностей. Его мать первой учила Эвариста вплоть до его 12-летия. Она учила его греческому, латинскому и религии, при этом она передала сыну свой религиозный скептицизм. Отец Галуа был важным человеком в обществе, и в 1815 г. он был избран мэром Бур-ля-Рен.

Определенно первым историческим событием, сыгравшим наиболее важную роль в жизни Галуа, был штурм Бастилии 14 июля 1789 г. С этого момента у монархии Людовика XVI возникли большие трудности, поскольку большинство французов уладили свои разногласия и объединились в попытке разрушить привилегированные учреждения церкви и государства.

Не пойдя на компромисс, Людовик XVI был осужден после попытки бежать из страны. После казни короля 21 января 1793 г. установилось царство террора со множеством политических процессов. К концу 1793 г. в Париже было уже 4595 политических заключенных. Позднее для Франции наступили времена получше, когда ее армии под командованием Наполеона Бонапарта одерживали одну победу за другой.

Наполеон в 1800 г. стал первым консулом, а затем, в 1804 г. императором. Французские армии продолжали покорение Европы, а вместе с этим власть Наполеона все более усиливалась. В 1811 г. власть Наполеона достигла своего пика. К 1815 г. правление Наполеона закончилось. За

[1] По БСЭ дата смерти — 30 мая. — *Прим. ред.*

разгромом в русской кампании следовали новые поражения, и 31 марта 1814 г. союзные войска вошли в Париж. 6 апреля Наполеон отрекся от престола, и союзники возвели на французский трон Людовик XVIII. В 1815 г. мир увидел знаменитые сто дней Наполеона. Он вошел в Париж 20 марта, был разбит под Ватерлоо 18 июня и вновь отрекся от престола 22 июня. Людовик XVIII вновь стал королем. Он умер в сентябре 1824 г., после чего новым королем стал Карл X.

Галуа в это время учился в школе. 6 октября 1823 г. он был принят в 4-й класс лицея Луи-ле-Гран. Во время первого семестра его учебы произошло небольшое восстание, и 40 учеников были исключены из школы. Галуа в этом не участвовал. В течение 1824–25 гг. его обучение шло успешно, и он даже получил несколько премий. Однако в 1826 г. ему предложили остаться на второй год, поскольку его успеваемость по риторике не достигала требуемого уровня.

Февраль 1827 г. стал поворотной точкой в жизни Галуа. Он был принят в свой первый математический класс — класс М. Вернье. Математика быстро его захватила, и вскоре руководитель его обучения писал:

Им завладела страсть к математике. Я думаю, что для него было бы лучше, если бы его родители позволили ему учиться одной только математике. Он тратит все свое время на это и не занимается ничем другим, тем самым мучая учителей и обрекая себя на наказания.

Школьные записи стали характеризовать Галуа как странного, эксцентричного, оригинального и замкнутого человека. Интересно, что большинство когда-либо живших выдающихся математиков подвергались критике за оригинальность. Однако М. Вернье сообщал о Галуа:

Смышленный, делающий успехи, но недостаточно последовательный.

В 1828 г. Галуа сдавал экзамены в Политехническую школу, но провалился. Это был ведущий университет Парижа, и Галуа хотел поступить туда по академическим соображениям. Кроме того, в то время в этом университете существовало сильное политическое движение, и Галуа, будучи по примеру своих родителей ярым республиканцем, желал в этом движении поучаствовать.

Вернувшись в Луи-ле-Гран, Галуа записался в математический класс Луи Ришара. Однако он все больше вел собственные исследования и все меньше времени посвящал школьным занятиям. Он изучил *"Геометрию"* Лежандра и трактаты Лагранжа. Как писал Ришар,

Этот студент работает только в высших сферах математики.

В апреле 1829 г. Галуа опубликовал свою первую работу о цепных дробях в "Математических хрониках". 25 мая и 1 июня он представил статьи о решениях алгебраических уравнений в Академию наук. Рецензентом этих статей был назначен Коши.

2 июля 1828 г. отец Галуа покончил с собой. Эта трагедия сильно ударила по сыну. Священник Бур-ля-Рен подделал имя мэра Галуа на злобных сфабрикованных эпиграммах, направленных против собственных родственников Галуа. Отец Галуа был человеком мягким, и гарантированный этими эпиграммами скандал был за пределами того, что он мог вынести. Он наложил на себя руки в своей парижской квартире всего в нескольких шагах от Луи-ле-Гран, где учился его сын. Смерть отца глубоко подействовала на Галуа и оказала огромное влияние на его дальнейшую жизнь.

Через несколько недель после смерти отца Галуа вновь попытался поступить в Политехническую школу и вновь неудачно. Он провалился вторично частью из-за того, что находился в наихудшей жизненной ситуации, связанной с недавней смертью отца, частью из-за того, что никогда не мог хорошо объяснить суть своих глубоких математических идей. Из-за этого провала Галуа пришлось поступить в Нормальную школу, которая была дополнением к Луи-ле-Гран, и при этом еще сдавать экзамены на бакалавра, которых он мог бы избежать, поступив в Политехническую школу.

Он сдал экзамены удачно и получил степень 29 декабря 1829 г. Его экзаменатор по математике сообщает:

Иногда этот ученик невразумительно выражает свои идеи, но он умен и демонстрирует замечательный дух исследования.

Его экзаменатор по литературе сообщает:

Это единственный студент, ответивший мне плохо, он абсолютно ничего не знает. Мне сказали, что у него экстраординарные способности к математике. Это меня крайне удивило, поскольку после экзамена я считал его не слишком умным.

Галуа послал Коши следующую работу по теории уравнений, но затем узнал из *"Бюллетеня де Ферюссак"* о посмертной статье Абеля, которая покрывала часть его работы. По совету Коши он подал в феврале 1830 г. на рассмотрение новую статью *"Об условиях разрешимости уравнений в радикалах"*. Статья была послана Фурье, секретарю академии, для рассмотрения на Гран При в математике. В апреле 1830 г. Фурье умер, статья была утеряна, поэтому она никогда не участвовала в конкурсе на этот приз.

После прочтения работ Абеля и Якоби Галуа работал в теории эллиптических функций и абелевых интегралов. При поддержке Жака Штурма он опубликовал три статьи в "Бюллетене де Феррюссак" в апреле 1830 г. Однако в июне он узнал, что приз академии будет присужден совместно Абелю (посмертно) и Якоби, а его собственная работа не рассматривалась.

В июле 1830 г. произошла революция. Карл X бежал из Франции. На улицах Парижа начались волнения, директор Нормальной школы, М. Гиньо, запер студентов, чтобы не допустить их участия в этом. Галуа

пытался перелезть через стену, чтобы принять участие в восстании, но не смог. В декабре 1830 г. М. Гиньо написал газетную статью с нападками на студентов. Галуа написал в *Школьной газете* ответ с обвинениями М. Гиньо за то, что тот запер студентов в школе. За это письмо Галуа был исключен и пошел на службу в артиллерию национальной гвардии, республиканскую часть милиции. В декабре 1830 г. артиллерия национальной гвардии была ликвидирована королевским указом, поскольку новый король Луи Филипп чувствовал, что она угрожает его трону.

Две маленьких публикации: тезисы в *"Хрониках Жергонна"* (декабрь 1830 г.) и письмо на тему научного обучения в *"Школьной газете"*, были последними в жизни Галуа. В январе 1831 г. Галуа попытался вернуться к математике. Он организовал математические классы по высшей алгебре, что привлекло на первое собрание 40 студентов. После этой встречи число студентов быстро уменьшилось. Пуассон предложил Галуа представить в академию третий вариант его статьи об уравнениях, и Галуа сделал это 17 января.

18 апреля Софи Жермен написала письмо своему другу — математику Либри, в котором описала положение Галуа.

... смерть М. Фурье слишком много значила для этого студента Галуа, который, несмотря на свою дерзость, демонстрирует признаки таланта. Все это так сильно повлияло, что он был исключен из Нормальной школы. Он остался без средств... Говорят, что он скоро сойдет с ума. Боюсь, что это правда.

В конце 1830 г. 19 офицеров артиллерии национальной гвардии были арестованы и обвинены в заговоре с целью свержения правительства. Они были оправданы, и 9 мая 1831 г. 200 республиканцев собрались на праздничный обед по случаю их оправдания. Во время обеда Галуа поднял свой бокал и с кинжалом в руке высказал угрозу в адрес короля Луи Филиппа. После обеда Галуа был арестован и брошен в тюрьму Сент-Пелажи. На суде, проходившем 15 июня, его адвокат объявил, что Галуа сказал:

Луи Филипп, если ты предашь...

а последние слова потонули в шуме. Галуа был оправдан, что весьма удивительно, поскольку он по существу повторил свою угрозу со скамьи подсудимых.

В день взятия Бастилии 14 июля Галуа был снова арестован. Он надел форму артиллерии национальной гвардии, которая была вне закона. Кроме того он носил заряженное ружье, несколько пистолетов и кинжал. Его снова отправили в тюрьму Сент-Пелажи. В тюрьме он узнал, что его статья отклонена. Пуассон сообщал:

Его аргументы недостаточно ясны и недостаточно развиты, чтобы мы могли считать их строгими.

Однако он выразил поддержку Галуа в деле публикации более полного доклада об этой работе. В тюрьме Сент-Пелажи Галуа пытался совершить самоубийство, заколов себя кинжалом, но другие заключенные помешали ему. Напившись в тюрьме, он излил свою душу:

Знаешь, как мне не хватает друга? Я только тебе это доверю: мне нужно полюбить кого-нибудь, но только всей душой. Я потерял отца, и никто мне его некогда не заменит, слышишь?

В марте 1832 г. эпидемия холеры поразила Париж и заключенных и Галуа в том числе. Он был переведен в больницу Сью Фолтрие. Там он влюбился в Стефани-Фелицию дю Мотель, дочь местного врача. После освобождения Галуа 29 апреля они со Стефани обменялись письмами, и стало ясно, что она пытается сохранить дистанцию.

Имя Стефани несколько раз появлялось на полях одной из рукописей Галуа.

Галуа дрался на дуэли с Пеше д'Эрбенвилем 30 мая. Смысл дуэли не до конца ясен, но определенно как-то связан со Стефани.

На полях своей рукописи в ночь перед дуэлью Галуа написал:

Требуется еще кое-что, чтобы закончить это доказательство. У меня нет на это времени. (Замечание автора).

Это породило легенду о том, что он провел свою последнюю ночь, записывая все, что знает о теории групп. Эта история кажется некоторым преувеличением.

Галуа был ранен на дуэли и брошен и д'Эрбенвилем, и своими секундантами. Его нашел крестьянин. Галуа умер 31 мая в госпитале Кошен. Его похороны состоялись 2 июня. Это стало поводом для сплочения республиканцев и волнений, продолжавшихся несколько дней.

Брат Галуа и его друг Шевалье скопировали его математические статьи и разослали их Гауссу, Якоби и другим. Желанием Галуа было то, чтобы Якоби и Гаусс высказали свои мнения о его работе. Нет никаких записей, свидетельствующих о том, что они это сделали. Однако статьи получил и Лиувилль. В сентябре 1843 г. он анонсировал в академии, что нашел в статьях Галуа сжатое решение

... верное и глубокое для этой милой задачи: как по данному неприводимому уравнению простой степени установить, имеет ли оно решение в радикалах.

Лиувилль опубликовал эти статьи Галуа в своем журнале в 1846 г.

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Galois.html

С.5 Иоганн Карл Фридрих Гаусс

Родился: 30 апреля 1777 г. в Брауншвейге, герцогство Брауншвейг (теперь Германия).

Умер: 23 февраля 1855 г. в Гёттингене, Ганновер (теперь Германия).

Карл Фридрих Гаусс работал в большом числе областей математики и физики, среди которых теория чисел, анализ, дифференциальная геометрия, геодезия, магнетизм, астрономия и оптика. Его работы оказали огромное влияние на многие области.

В возрасте семи лет Карл Фридрих начал учиться в начальной школе, и почти сразу его потенциал был замечен. Его учитель, Бюттнер вместе со своим ассистентом Мартином Бартелсом были изумлены тем, как он мгновенно сложил целые числа от 1 до 100, разбив их на 50 пар, в каждой из которых сумма равна 101.

В 1788 г. Гаусс при помощи Бюттнера и Бартелса начал учиться в гимназии, где он обучался немецкому и латыни. После получения стипендии от герцога Брауншвейга — Вольфенбюттеля, Гаусс поступил в 1792 г. в королевский колледж Брауншвейга. В академии Гаусс независимо открыл закон Боде, биномиальную теорему, и неравенство между средним геометрическим и средним арифметическим, а также закон квадратичной взаимности и теорему о простых числах.

В 1795 г. Гаусс покинул колледж Брауншвейга, чтобы учиться в Гёттингенском университете. Там учителем Гаусса был Кестнер, которого Гаусс часто осмеивал. Единственным известным его другом среди студентов был Фаркаш Бойяи. Они встретились в 1799 г. и многие годы переписывались друг с другом.

В 1798 г. Гаусс покинул Гёттинген без диплома. Но в это время он сделал одно из наиболее важных своих открытий — способ построения правильного 17-угольника с помощью циркуля и линейки. Это было главное достижение в этой области со времен греческих математиков. Оно было опубликовано как глава VII известной работы Гаусса *"Арифметические исследования"*.

В 1799 г. Гаусс вернулся в колледж Брауншвейга, где получил степень. После того как герцог Брауншвейга согласился продолжить выплату стипендии Гауссу, он попросил, чтобы Гаусс представил свою докторскую диссертацию в университет Хелмстедта. К тому времени Гаусс уже был знаком с Пфаффом, который был выбран на роль его научного руководителя. В диссертации Гаусса обсуждались фундаментальные теоремы алгебры.

Благодаря поддерживающей его стипендии, у Гаусса не было необходимости в поиске работы, и он мог целиком посвятить себя исследованиям. Летом 1801 г. он опубликовал книгу *"Арифметические исследования"*. В ней было семь глав и все, за исключением последней, упоминавшейся выше, были посвящены теории чисел.

В июне 1801 г. астроном Зах, с которым Гаусс познакомился за два или три года до этого, опубликовал орбитальные положения Цереры, нового астероида, который был открыт 1 января 1801 г. итальянским астрономом Г. Пиацци. К несчастью, Пиацци имел возможность наблюдать только 9 градусов ее орбиты перед тем, как она скрылась за Солнцем. Зах опубликовал несколько предсказаных им ее положений, в том числе и одно, сильно отличающееся от остальных, предсказанное ранее Гауссом. 7 декабря 1801 г. Зах переоткрыл Цереру, наблюдая ее почти в том месте, где предсказывал Гаусс. Гаусс использовал для этого предсказания аппроксимацию своим методом наименьших квадратов, хотя в то время он еще не раскрывал этот способ.

В июне 1802 г. Гаусс посетил Олберса, открывшего в марте того же года Паллас, и исследовал орбиту этого небесного тела. Олберс попросил Гаусса стать директором новой обсерватории Гёттингена, но этого не произошло. Гаусс начал переписываться с Бесселем, которого он не встречал вплоть до 1825 г., а также с Софи Жермен.

9 октября 1805 г. Гаусс женился на Джоанне Остофф. Так счастливо начавшаяся его личная жизнь была омрачена, когда его благодетель — герцог Брауншвейга — погиб, сражаясь в прусской армии. В 1807 г. Гаусс покинул Брауншвейг, чтобы занять должность директора Гёттингенской обсерватории.

Гаусс приехал в Гёттинген в конце 1807 г. В 1808 г. умер его отец, а годом позже умерла жена Гаусса Джоанна после родов их второго сына, который также умер вскоре после нее. Гаусс был совершенно разбит и написал Олберсу, попросив отпустить его домой на несколько недель,

чтобы собрать новую силу в руках — силу для жизни, которая ценна только потому, что принадлежит трем моим маленьким детям.

На следующий год Гаусс женился во второй раз на лучшей подруге Джоанны — Минне, и хотя у них было трое детей, кажется, что эта женитьба вполне подходила Гауссу.

Работе Гаусса его личные трагедии никогда не вредили. В 1809 г. он опубликовал свою вторую книгу, *"Theoria motus corporum coelestium*

sectionibus conicis Solem ambientium" ("Теория движения небесных тел") — главный двухтомный трактат о движении небесных тел. В первом томе он обсуждал дифференциальные уравнения, конические сечения и эллиптические орбиты, а во втором томе — главной части работы — он показал, как приближенно рассчитывать орбиты планет, а затем уточнять приближение. После 1817 г. работа Гаусса в теоретической астрономии прекратилась, хотя он продолжал вести наблюдения вплоть до 70-летнего возраста.

Много времени Гаусс тратил на новую обсерваторию, строительство которой было завершенно к 1816 г., но он все-таки находил время для работы в других областях. В это время он опубликовал *"Disquisitiones generales circa seriem infinitam"* — строгое обращение с рядами и введение в гипергеометрические функции, *"Methodus nova integralium valores per approximatonem inveniendi"* — практический очерк о приближенном интегрировании, *"Bestimmung der Genauigkeit der Beobachtungen"* — обсуждение методов статистических оценок, и *"Theoria attractionis corporum sphaeroidicorum ellipticorum homogeneorum methodus nova tractata"*. Последняя работа была вызвана геодезическими проблемами и касалась в основном теории потенциала. В действительности, в 1820-х гг. Гаусс все более и более интересовался геодезией.

В 1818 г. ему предложили произвести геодезическую съемку королевства Ганновер и связать ее с уже существующей картой Дании. Он с радостью принял это предложение и начал самостоятельную работу. Днем он проводил измерения, а ночью их обрабатывал, используя свои выдающиеся умственные способности к вычислениям. Он постоянно писал Шумахеру, Олберсу и Бесселю, рассказывая о своих продвижениях и обсуждая проблемы.

В процессе этой съемки Гаусс изобрел гелиотроп, работа которого была основана на отражении солнечных лучей с использованием системы зеркал и маленького телескопа. Однако при съемке использовались неточная основа для линий и неудовлетворительная сеть треугольников. Гаусс часто удивлялся, если ему советовали продолжить другие его занятия, но при этом он опубликовал свыше 70 статей между 1820 и 1830 г.

В 1822 г. Гаусс получил приз копенгагенского университета за *"Theoria attractionis..."* и идею отображения одного пространства на другое с сохранением локального подобия. Эта статья была опубликована в 1825 г. и привела к значительно более поздней публикации *"Untersuchungen über Genenstände Högeren Geodäsie"* (1843 и 1846 гг.). Статья *"Theoria combinationis observatorium erroribus minimus obnoxiae"* (1823 г.) с дополнением (1828 г.) посвящена математической статистике, и, в частности, методу наименьших квадратов.

С начала 1800-х гг. Гаусс заинтересовался вопросом о возможности существования неэвклидовой геометрии. Он подробно обсуждал эту тему с Фаркашем Бойяи и в своей переписке с Герлингом и Шумахером. В 1816 г. в литературном обзоре он обсуждал доказательства выводимости

аксиомы параллельности из остальных акиом Эвклида. Возможно, что он предполагал существование неэвклидовых геометрий, хотя и весьма неуверенно. Он доверился Шумахеру, сообщив ему, что боится за свою репутацию в том случае, если публично признает, что верит в существование такой геометрии.

В 1831 г. Фаркаш Бойяи прислал Гауссу работу своего сына Яноса Бойяи на эту тему. Гаусс ответил

хвалить эту работу — означало бы хвалить себя.

Десятилетием позже, когда ему сообщили о работе Лобачевского на эту тему, он похвалил ее "подлинно геометрический" характер, а в письме Шумахеру в 1846 г., утверждал, что он

54 *года был убежден в этом,*

показав тем самым, что он знал о существовании неэвклидовой геометрии с 15-летнего возраста (это кажется невероятным).

У Гаусса был большой интерес к дифференциальной геометрии. Он опубликовал множество статей в этой области. *"Disquisitiones generales circa superficies curva"* (1828 г.) была наиболее знаменитой его работой на эту тему. На самом деле эта статья возникла из его геодезических интересов, но в ней содержатся и геометрические идеи, в частности, идея гауссовой кривизны. Эта статья также включает в себя "Theorema Egregium" Гаусса:

Если область из E^3 можно изометрично отобразить в другую область из E^3, то значения гауссовой кривизны в соответствующих точках будут одинаковы.

Период 1817–1832 гг. был особенно горестным временем для Гаусса. В 1817 г. он взял к себе свою больную мать, и она жила с ним вплоть до своей смерти в 1839 г. Все это время он спорил со своей женой и ее родственниками о том, стоит ли им переехать жить в Берлин. Ему предложили место в Берлинском университете, и Минна и ее родственники сильно хотели переехать в Берлин. Однако Гаусс не любил перемен и решил остаться Гёттингене. В 1831 г. вторая жена Гаусса умерла после длительной болезни.

В 1831 г. в Гёттингене появился Вильгельм Вебер, он стал профессором физики кафедры Тобиаса Майера. Гаусс знал Вебера с 1828 г. и поддержал его назначение. Гаусс работал в области физики до 1831 г., опубликовав *"Uber ein neues allgemeines Grungesetz der Mechanik"*, где содержится принцип наименьшего принуждения, и *"Principia generalia theoriae figurae fluidorum in statu aequilibrii"*, где обсуждаются силы тяготения. Эти статьи основаны на гауссовой теории потенциала, продемонстрировавшей свою огромную важность в его физических работах. Позднее Гаусс пришел к убеждению в том, что его теория потенциала и его метод наименьших квадратов обеспечивают жизненную связь между наукой и природой.

В 1832 г. Гаусс и Вебер начали исследования теории земного магнетизма после того, как Александр фон Гумбольдт обратился к Гауссу за помощью в создании на Земле сети пунктов наблюдения за магнитными явлениями. Гаусс был захвачен этим проектом, и к 1840 г. он написал три важных статьи на эту тему: *"Intensitas vis magneticae terrestris ad mensuram absolutam revocata"* (1832 г.), *"Allgemeine Theorie des Erdmagnetismus"* (1839 г.) и *"Allgemeine Lehrsätze in Beziehung auf die im verkehtren Verhältnisse des Quadrats der Entfernung wirkenden Anziehungs- und Abstossungskräfte"* (1840 г.). Все эти статьи были посвящены теориям земного магнетизма того времени, включая идеи Пуассона, абсолютное измерение магнитной силы и эмпирическое определение земного магнетизма. Принцип Дирихле был упомянут без доказательства.

В работе *"Allgemeine Theorie..."* показано, что земной шар может иметь только два полюса, и продолжено доказательство важной теоремы, касающейся определения интенсивности горизонтальной компоненты магнитной силы вместе с углом наклона. Гаусс применил уравнение Лапласа, которое помогло ему в его вычислениях, и завершил определение точного местоположения южного магнитного полюса.

Гумбольдт изобрел календарь для наблюдений за магнитным склонением. Когда была построена новая магнитная обсерватория Гаусса (стройка завершена в 1833 г. без применения каких-либо магнитных металлов), Гаусс видоизменил многие из процедур Гумбольдта, что тому очень не понравилось. Однако внесенные Гауссом изменения позволили ему получать более точные результаты с меньшим числом ошибок.

За шесть лет Гаусс и Вебер вместе многого достигли. Они открыли законы Кирхгофа, построили примитивный телеграф, который мог пересылать сообщения на расстояние в 1.5 километра. Однако для Гаусса все это было лишь развлечением. Его более интересовала задача создания всемирной сети точек наблюдения за магнитными явлениями. Это занятие привело ко многим конкретным результатам. Было основано общество *Magnetischer Verein* и его журнал. Был издан атлас геомагнетизма. Гаусс и Вебер издавали и собственный журнал, в котором публиковали собственные результаты с 1836 по 1841 гг.

В 1837 г. Вебер был вынужден покинуть Гёттинген, после того как он вмешался в политический спор, и с этого момента активность Гаусса начала постепенно снижаться. Он продолжал писать письма в ответ на открытия своих ученых коллег, при этом обычно добавляя, что методы ему давно известны, но он не чуствовал необходимости их публиковать. Иногда он с большой похвалой отзывался о продвижениях, сделанных другими математиками, в частности, Эйзенштейном и Лобачевским.

С 1845 по 1851 гг. Гаусс работал в фонде вдовцов Гёттингенского университета. Эта работа дала ему практический опыт проведения финансовых операций, и он продолжал успешно инвестировать средства в финансовые обязательства частных компаний.

Двумя последними докторантами Гаусса были Морис Кантор и Дедекинд. Дедекинд оставил прекрасное описание своего научного руководителя

... обычно он сидел в удобной позе, глядя вниз, слегка сутулясь, со сложенными на коленях руками. Он говорил совершенно свободно, очень ясно, просто и понятно, но если он хотел выделить новую точку зрения ..., то он поднимал голову, поворачивался к одному из тех, кто был рядом, и пристально смотрел на него пронзительным взглядом своих голубых глаз, произнося выразительную речь. ... Если он переходил от объяснения принципов к написанию математических формул, то он вставал и, стоя абсолютно прямо, писал на доске, расположенной позади него, своим особенным великолепным почерком. При этом он достигал нужной ему цели экономно и продуманно, используя довольно мало места. Для корректного приведения численных примеров он приносил нужные данные на маленьких полосках бумаги.

В 1849 г. Гаусс прочитал лекцию в честь своего золотого юбилея — пятидесятилетия с момента, когда за его дипломную работу ему был предоставлен грант университета Хелмстедта. Лекция посвящалась изменениям в его диссертации 1799 г. Из математического сообщества присутствовали только Якоби и Дирихле, но Гаусс получил множество поздравлений.

С 1850 г. деятельность Гаусса вновь почти целиком лежала в практической плоскости, хотя он улучшил докторскую диссертацию Римана и выслушал его пробную лекцию. Последняя его известная научная переписка — это переписка с Герлингом. В 1854 г. они обсуждали модифицированный маятник Фуко. Он также занимался вопросом открытия новой железнодорожной связи между Ганновером и Гёттингеном, но это был его последнии выход в общество. Его здоровье медленно ухудшалось, и ранним утром 23 февраля 1855 г. Гаусс умер во сне.

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Gauss.html

С.6 Карл Густав Якоб Якоби

Родился: 10 декабря 1804 г. в Потсдаме, Пруссия (теперь Германия).
Умер: 18 февраля 1851 г. в Берлине, Германия.

Карл Якоби основал теорию эллиптических функций.

Якоби рос в преуспевающей семье банкира. Он получил хорошее образование в Берлинском университете. В 1825 г. он получил степень доктора философии. С 1826 г. до своей смерти Якоби преподавал математику в университете Кенигсберга, в 1832 г. он был назначен на кафедру.

Он разработал теорию эллиптических функций, основанную на четырех тета-функциях. Его работа 1829 г. *"Fundamenta nova theoria functionum ellipticarum"* и более поздние дополнения были основным вкладом в теорию эллиптических функций.

В 1834 г. Якоби доказал, что если однозначная функция от одной переменной дважды периодична, то отношение периодов чисто мнимое. Этот результат указал направление для последующих работ в этой области, в частности, Лиувиллю и Коши.

Якоби провел важное исследование дифференциальных уравнений первого порядка в частных производных и применил результат к дифференциальным уравнениям динамики.

Он также работал с определителями и изучал функциональный определитель, теперь называемый якобианом. Якоби не был первым, кто изучал функциональный определитель, носящий теперь его имя, впервые он появился в статье Коши 1815 г. Якоби же в 1841 г. написал монографию *"De determinantibus functionalibus"*, посвященную этому определителю. Среди прочего он доказал, что если n функций от n переменных функционально зависимы, то якобиан тождественно равен нулю, а если функции независимы, то якобиан не может быть тождественно равен нулю.

Репутация великолепного преподавателя привлекала к Якоби многих студентов. Он ввел метод семинаров для обучения студентов последним достижениям в математике.

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Jacobi.html

С.7 Адриен-Мари Лежандр

Родился: 18 сентября 1752 г. в Париже, Франция.

Умер: 10 января 1833 г. в Париже, Франция.

Основная работа **Лежандра** об эллиптических интегралах обеспечила математическую физику базовыми аналитическими инструментами.

Лежандр учился в колледже Мазарини в Париже. С 1775 по 1780 гг. он преподавал вместе с Лапласом в Военной школе, куда был назначен по совету д'Аламбера. В 1783 г. Лежандр был назначен в Академию наук и работал там вплоть до ее закрытия в 1793 г.

В 1782 г. Лежандр определил силу притяжения для некоторых тел вращения, введя бесконечную серию многочленов P_n, теперь называемых многочленами Лежандра.

Его главные результаты об эллиптических функциях в работе *"Упражнения по интегральному исчислению"* (1811, 1817, 1819 гг.) и эллиптических интегралах в работе *"Трактат об эллиптических функциях"* (1825, 1826, 1830 гг.) предоставили математической физике базовые аналитические инструменты.

В своем известном учебнике *"Элементы геометрии"* (1794 г.) Лежандр дал простое доказательство иррациональности числа π, кроме того там он впервые доказал, что число π^2 тоже иррациональное, и высказал предположение, что оно не является даже алгебраическим, то есть не является корнем никакого алгебраического уравнения конечной степени с рациональными коэффициентами.

Более 40 лет он пытался доказать аксиому параллельности Эвклида.

В 1824 г. Лежандр отказался голосовать за правительственного кандидата для Национального института. Из-за этого ему прекратили выплату пенсии, и он умер в бедности. В октябре 1826 г. Абель писал:

Лежандр — черезвычайно любезный человек, но к сожалению он стар, как камни.

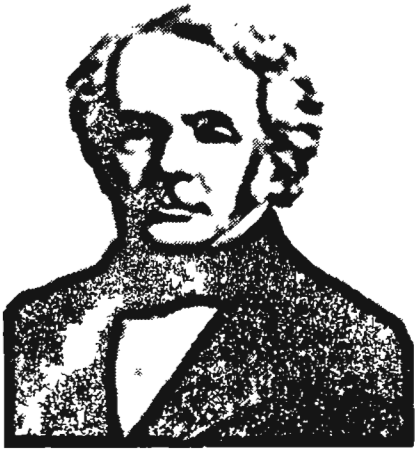
Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Legendre.html

С.8 Август Фердинанд Мёбиус

Родился: 17 ноября 1790 г. в Шульпфорта, Саксония (теперь Германия).

Умер: 26 сентября 1868 г. в Лейпциге, Германия.

Август Мёбиус более всего известен своими работами в топологии, особенно представлением об односторонней двумерной поверхности — ленте Мёбиуса.



Август был единственным ребенком Иоганна Генриха Мёбиуса — учителя танцев, умершего, когда Августу было три года. Его мать была потомком Мартина Лютера. До 13 лет Мёбиус учился дома, а затем в 1803 г., уже демонстрируя интерес к математике, он поступил в колледж в Шульпфорта.

В 1809 г. Мёбиус окончил колледж и стал студентом университета города Лейпцига. Его семья желала, чтобы он изучал юриспруденцию, и сначала он так и делал. Однако вскоре он понял, что этот предмет ему не нравится, и в середине первого года обучения он решил действовать в соответствии с собственными предпочтениями, а не желаниями родственников. Поэтому он взялся за изучение математики, астрономии и физики.

Преподавателем, оказавшим на Мёбиуса наибольшее влияние во время учебы в Лейпциге, был преподаватель астрономии Карл Молвейде. Хотя Молвейде и был астрономом, он хорошо известен многими математическими открытиями, среди которых, в частности, тригонометрические соотношения Молвейде, открытые в 1807–09 гг., и конформная, то есть сохраняющая углы, картографическая проекция Молвейде.

В 1813 г. Мёбиус посетил Гёттинген, где он изучал астрономию под руководством Гаусса. Тогда Гаусс уже был директором обсерватории Гёттингена, и, разумеется, величайшим математиком того времени. Таким образом, Мёбиус снова учился у астронома, интересующегося математикой. Из Гёттингена Мёбиус прибыл в Галл, где учился у Иоганна Пфаффа, учителя Гаусса. Под руководством Пфаффа он занимался математикой несколько больше, чем астрономией. Таким образом, на этом этапе Мёбиус много работал в обеих областях.

В 1815 г. Мёбиус закончил свою докторскую диссертацию на тему *"Затмения неподвижных звезд"* и начал работу над новой диссертацией. Известно, что во это время его пытались призвать в прусскую армию. Мёбиус писал:

Это самая ужасная идея, услышанная мной, и каждый, кто рискнет, отважится и наберется смелости мне это предложить, не спасется от моего кинжала.

Он избежал службы в армии и завершил свою новую диссертацию на тему *"Тригонометрические уравнения"*. Интерес Молвейде к математике был таким, что он перешел с кафедры астрономии на кафедру математики, таким образом, у Мёбиуса появилась реальная возможность стать профессором астрономии. И действительно, в 1816 г. он был назначен на кафедру астрономии и высшей механики Лейпцигского университета. Первоначально он получил назначение на должность экстраординарного профессора, хотя был довольно молод для такого назначения.

Однако Мёбиус не скоро стал полным профессором. Кажется, что он был не особенно хорошим лектором, и это делало его жизнь трудной, поскольку он не мог привлечь на свои лекции студентов, оплачивающих обучение. Он был вынужден объявить свои лекционные курсы бесплатными, пока студенты считают его курсы заслуживающими внимания.

В 1816 г. Мёбиусу предложили должность астронома в Грейфсвальде, а в 1819 г. — должность математика в Дорпате. Он отказался от обоих предложений, частично из-за своей веры в высокий уровень Лейпцигского университета, частично из-за своей верности Саксонии. В 1825 г. умер Молвейде, и Мёбиус надеялся перейти на кафедру математики, повторяя, как и раньше, путь Молвейде. Но этого не произошло, и на освободившееся место был назначен другой математик.

В 1844 г. репутация Мёбиуса, как исследователя, послужила причиной приглашения из университета города Иена. В это время университет Лейпцига вполне заслуженно сделал его полным профессором астрономии.

Со времени своего первого назначения в Лейпцигском университете Мёбиус состоял в должности наблюдателя Лейпцигской обсерватории. Он принимал участие в ее перестройке, а с 1818 по 1821 гг. надзирал за этим проектом. Перед тем, как высказать свои рекомендации по поводу новой обсерватории, он посетил несколько других обсерваторий Германии. В 1820 г. он женился. У него было трое детей: двое сыновей и дочь. В 1848 г. он стал директором обсерватории.

В 1844 г. Мёбиуса посетил Грассман. Он попросил Мёбиуса дать рецензию на его главную работу *"Die lineale Ausdehnundslehre, ein neuer Zweig der Mathematik"* (1844 г.), содержавшую много результатов, подобных результатам Мёбиуса. Однако Мёбиус не понял значения работы Грассмана и не стал ее рецензировать. Тем не менее он убедил Грассмана представить работу на конкурс, и после того, как Грассман выиграл приз, Мёбиус в 1847 г. написал рецензию.

Хотя наиболее известны работы Мёбиуса по математике, он публиковал важные работы и по астрономии. Он написал *"De Computandis Occultationibus Fixarium per Planetas"* (1815 г.) о затмениях планет. Кроме того он написал *"Die Hauptsätze der Astronomie"* (1836 г.) об основах астрономии и *"Die Elemente der Mechanik des Himmels"* (1843 г.) о небесной механике.

Математические публикации Мёбиуса были, может, и не всегда оригинальны, но эффективны и ясно изложены. Его вклад в математику так описан его биографом Ричардом Бальцером:

Вдохновение для своих исследований он находил, в основном, в богатом источнике своего оригинального ума. В его интуиции, в задачах, которые он себе ставил, и в решениях, которые он находил, во всем проявлялась какая-то сверхъестественная изобретательность, нечто оригинальное в каждом ранее известном методе. Он работал спокойно и без спешки. Его работа практически останавливалась до тех пор, пока каждая вещь не встанет на свое место. Без натиска, без помпезности и высокомерия он ожидал, пока не созреют плоды его ума. Только после такого ожидания он публиковал свои совершенные работы...

Почти все работы Мёбиуса были опубликованы в *"Журнале Крелля"* — первом журнале, посвященном исключительно математическим статьям. Работа Мёбиуса 1827 г. по аналитической геометрии *"Der barycentrische Calkul"* стала классической и включает множество его результатов в проективной и афинной геометрии. В ней он ввел однородные координаты и обсудил геометрические преобразования, в частности, проективные трансформации. Он ввел структуру, называемую теперь сетью Мёбиуса, игравшую важную роль в развитии проективной геометрии.

Имя Мёбиуса прикрепилось ко многим математическим объектам, таким как "функция Мёбиуса", которую он ввел в статье 1831 г. *"Uber eine besondere Art von Umkehrung der Reihen"*, и "формула обращения Мёбиуса".

В 1837 г.он опубликовал работу *"Lehrbuch der Statik"*, давшую геометрическую трактовку статики. Она инициировала изучение систем линий в пространстве.

Перед появлением вопроса Франца Гутри о раскраске карты в четыре цвета Мёбиус в 1840 г. сформулировал следующую более простую задачу.

У короля было пять сыновей. Повелел он, чтобы после его смерти королевство было разделено его сыновьями на пять областей так, чтобы каждая область имела общую границу с четырьмя остальными. Можно ли выполнить его волю?

Ответ, конечно, отрицательный, что легко показать. Но это демонстрирует интерес Мёбиуса к идеям в топологии — области, пионером которой он считается. В монографии, представленной в академию наук и единственной, обнаруженной после его смерти, он обсуждал свойства односторонних поверхностей, в том числе и ленты Мёбиуса, которую он открыл в 1858 г. Это открытие было сделано Мёбиусом, когда он работал над вопросом создания геометрической теории многогранников, поставленным Парижской академией.

На самом деле первым, кто описал объект, называемый теперь лентой Мёбиуса, был не Мёбиус, а Листинг, по каким бы критериям не оценивать первенство, хоть по дате публикации, хоть по дате первого открытия.

Лента Мёбиуса — это двумерная поверхность с одной стороной. Ее можно сконструировать в трех измерениях следующим образом. Возьмем прямоугольную бумажную ленту и соединим ее концы, повернув один на 180 градусов. Теперь можно, начав с точки A, прочертить на поверхности путь, проходящий через точку, лежащую, казалось бы, на другой стороне полоски бумаги относительно A.

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Mobius.html

С.9 Джозеф Генри Маклаген Веддербарн

Родился: 2 февраля 1882 г. в Форфар, Ангус, Шотландия.

Умер: 9 октября 1948 г. в Принстоне, Нью Джерси, США.

Джозеф Веддербарн сделал важные продвижения в теории колец и алгебр, а также в теории матриц.

В 1898 г. он поступил в Эдинбургский университет, а в 1903 г. получил степень по математике. Веддербарн продолжил обучение в Германии, проведя 1903–1904 гг. в университете Лейпцига, а затем семестр в Берлинском университете.

Он получил стипендию Карнеги для обучения в США и провел 1904–1905 гг. в Чикагском университете, где работал вместе с Вебленом. Вернувшись в Шотландию, он проработал 4 года в Эдинбурге ассистентом Георга Кристала. С 1906 по 1908 гг. он работал редактором Трудов Эдинбургского математического общества.

В 1909 г. Веддербарн был назначен на должность наставника по математике в Принстоне, где он присоединился к Веблену. С началом первой мировой войны Веддербарн пошел добровольцем в британскую армию и служил, в том числе и во Франции, до самого конца войны.

По возвращении в Принстон он скоро продвинулся по службе и в 1921 г. получил постоянную должность. С 1912 по 1928 гг. он был редактором журнала “Анналы математики”. Кажется, что с конца этого

периода Веддербарн страдал слабым нервным расстройством и стал более уединенным.

Лучшая работа Веддербарна была выполнена до его службы в армии. В 1905 г. он показал, что не существует конечных некоммутативных тел. Из этого следует полное описание структуры всех конечных проективных геометрий и то, что во всех этих геометриях теорема Паскаля следует из теоремы Дезарга.

В 1907 г. он опубликовал, вероятно, самую известную статью по классификации полупростых алгебр. Он показал, что каждая полупростая алгебра является прямой суммой простых, а каждая простая алгебра есть матричная алгебра над кольцом с делением (телом).

В общей сложности он опубликовал около 40 работ, в основном, о кольцах и матрицах. Наиболее известная его книга — *"Лекции о матрицах"* (1934 г.).

Сведения взяты на web-странице:
http://www-history.mcs.st-and.ac.uk/history/Mathematicians/Wedderburn. html

Приложение D

Новые функции

☐ AddTwoLetters

AddTwoLetters складывает две буквы по модулю 26, где $a = 0, b = 1, \ldots, z = 25$.

```
AddTwoLetters[a_, b_] :=
FromCharacterCode[Mod[(ToCharacterCode[a] - 97) +
    (ToCharacterCode[b] - 97), 26] + 97]
```

Пример:

```
PAddTwoLetters["b", "c"]
```

```
|| d
```

☐ CaesarCipher

Шифр Цезаря применяется с заданным ключом к заданному открытому тексту из строчных букв:

```
CaesarCipher[plaintext_, key_] :=
    FromCharacterCode[
  Mod[ToCharacterCode[plaintext] - 97 + key, 26] + 97]
```

Пример:

```
plaintext = "typehereyourplaintextinsmallletters";
key = 24;
CaesarCipher[plaintext, key]
```

```
|| rwncfcpcwmspnjyglrcvrglqkyjjjcrrcpq
```

☐ ColumnSwap

ColumnSwap меняет местами столбцы в матрице.

```
ColumnSwap[B_, i_, j_] := Module[{U, V}, U = Transpose[B];
V = U[[i]], U[[i]] = U[[j]], U[[j]] = V; Transpose[U]]
```

Пример:

```
Clear[a, b, ,c, d, e, f, g, h, i, j, k, l];
```

$$\texttt{A} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{pmatrix}$$

```
AA = ColumnSwap[A, 1, 4];
MatrixForm[AA]
```

$$\begin{pmatrix} d & b & c & a \\ h & f & g & e \\ l & j & k & i \end{pmatrix}$$

□ CoPrimeQ

CoPrimeQ проверяет, являются ли два данных числа взаимно простыми, т.е. будет ли их НОД равен 1.

```
CoPrimeQ[n_Integer, m_Integer] := GCD[n, m] == 1
```

Пример:

```
CoPrimeQ[35, 91]
CoPrimeQ[36, 91]
```

```
False
True
```

□ CoPrimes

CoPrimes генерирует список всех целых чисел между 1 и n, которые взаимно просты с n. Другими словами, она генерирует приведенную систему вычетов по модулю n.

CoPrimes использует определенную выше функцию CoPrimeQ.

```
CoPrimes[n_Integer ? Positive] :=
    Select[Range[n], CoPrimeQ[n, #] &]
```

Пример:

```
CoPrimes[15]
```

```
{1, 2, 4, 7, 8, 11, 13, 14}
```

□ DivisorProduct

DivisorProduct вычисляет $\prod_{d|n} f(d)$.

```
DivisorProduct[f_, n_] := Times @@ (f / @ Divisors[n])
```

Пример:

```
f[n_] := n;
DivisorProduct[f, 25]
```

```
|| 125
```

☐ DivisorSum

DivisorSum вычисляет $\sum_{d|n} f(d)$.

```
DivisorSum[f_, n_] := Plus @@ (f / @ Divisors[n])
```

Пример:

```
f[n_] := n;
DivisorSum[f, 15]
```

```
|| 24
```

☐ EllipticAdd

EllipticAdd вычисляет сумму двух точек на эллиптической кривой над $\mathbb{Z}_p$, заданной уравнением $y^2 = x^3 + a \cdot x^2 + b \cdot x + c$. Здесь p — простое, $p > 2$.

```
EllipticAdd[p_, a_, b_, c_, P_List, Q_List] :=
Module[{lam, x3, y3, R},
  Which[P == {0}, R = Q,
    Q == {0}, R = P,
    P[[1]] != Q[[1]],
      lam = Mod[
    (Q[[2]] - P[[2]])*PowerMod[Q[[1]] - P[[1]], -1, p], p];
      x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
      y3 = Mod[-(lam * (x3 - P[[1]]) + P[[2]]), p];
      R = {x3, y3},
    (P == Q) && (P != {0}),
      lam = Mod[(3 * P[[1]]^2 + 2 * a * P[[1]] + b) *
    PowerMod[2 * P[[2]], -1, p], p];
      x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
      y3 = Mod[-(lam * (x3 - P[[1]]) + P[[2]]), p];
      R = {x3, y3},
    (P[[1]] == Q[[1]]) && (P[[2]] != Q[[2]]), R = {0}];
  R]
```

Пример:

```
p = 11; a = 0; b = 6; c = 3;
EllipticAdd[p, a, b, c, {4, 6}, {9, 4}]
```

```
|| {3, 9}
```

☐ Entropy

Вычисляется функция энтропии $-p \cdot \log_2 p - (1-p)\log_2(1-p)$.

```
Entropy[p_] := -p * Log[2, p] -(1 - p) * Log[2, 1 - p];
```

Пример:

```
Entropy[1 / 2]
```

```
|| 1
```

☐ ListQuadRes

ListQuadRes дает список всех квадратичных вычетов по модулю p.

```
ListQuadRes[p_] :=
  Select[Range[p], JacobiSymbol[#1, p] == 1 &]
```

Пример:

```
p = 17;
ListQuadRes[p]
```

```
|| {1, 2, 4, 8, 9, 13, 15, 16}
```

☐ MultiEntropy

MultiEntropy вычисляет $-\sum_{i=1}^{n} p_i \log_2 p_i$ для списка $\{p_1, p_2, \ldots, p_n\}$.

```
MultiEntropy[p_List] := -  Sum[(p)p[[i]] * Log[2, p[[i]]], {i, 1, Length[p]}]
```

Пример:

```
p = {1 / 4, 1 / 4, 1 / 4, 1 / 4};
MultiEntropy[p]
```

```
|| 2
```

☐ MultiplicativeOrder

MultiplicativeOrder вычисляет мультипликативный порядок целого числа a по модулю n в предположении, что они взаимно просты. Таким образом, на выходе — наименьшее натуральное число m, такое, что $a^m \equiv 1 \pmod{n}$.

```
MultiplicativeOrder[a_, n_] := If[GCD[a, n] == 1,
    Divisors[EulerPhi[n]] //.
    {x_, y_} -> If[PowerMod[a, x, n] == 1, x, {y}]];
```

Пример:

```
MultiplicativeOrder[2, 123456789123]
```

```
|| 1285901112
```

☐ KnapsackForSuperIncreasingSequence

KnapsackForSuperIncreasingSequence находит $\{0,1\}$-решение проблемы рюкзака $\sum_{i=1}^{n} x_i \cdot a_i = S$, где $\{a_i\}_{i=1}^{n}$ — сверхвозрастающая последовательность.

```
KnapsackForSuperIncreasingSequence[a_List, S_] :=
    Module[{n, x, X, T},
        n = Length[a]; X = {}; T = S;
        While[n≥ 1,
            If[T≥a[[n]], x = 1, x = 0];
            T = T - x * a[[n]];
            X = Join[{x}, X]; n = n - 1];
        If[T != 0, Print["No solution"], X]]
```

Пример:

```
a = {22, 89, 345, 987, 4567, 45678}; S = 5665;
X = KnapsackForSuperIncreasingSequence[a, S]
```

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

☐ RowSwap

RowSwap меняет местами строки в матрице.

```
RowSwap[B_, i_, j_] := Module[{U, V}, U = B;
  V = U[[i]]; U[[i]] = U[[j]]; U[[j]] = V; U];
```

Пример:

```
Clear[a, b, c, d, e, f, g, h, i, j, k, l];
```

$$\mathtt{A} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{pmatrix};$$

```
AA = RowSwap[A, 1, 4];
MatrixForm[AA]
```

$$\begin{pmatrix} j & k & l \\ d & e & f \\ g & h & i \\ a & b & c \end{pmatrix}$$

Список литературы

[Adle79] Adleman L.M. *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, in Proc IEEE 20-th Annual Symp. on Found. of Comp. Science, pp. 55–60, 1979.

[Adle83] Adleman L.M. *On breaking the iterated Merkle – Hellman public key cryptosystern*, in Proc. 15-th Annual ACM Symp. Theory of Computing, pp. 402–412, 1983.

[Adle94] Adleman L.M. *The function field sieve*, Lecture Notes in Computer Science 877, Springer Verlag, Berlin, etc., pp. 108–121, 1995.

[AdDM93] Adleman L.M., J. DeMarrais, *A subexponential algorithm for discrete logarithms over all finite fields*, Mathematics of Computation, 61, pp. 1–15, 1993.

[AdPR83] Adleman L.M., C. Pomerance, R. Rumely, *On distinguishing prime numbers from composite numbers*, Annals of Math., 17, pp. 173–206, 1983.

[Aign79] Aigner M. *Combinatorial Theory*, Springer Verlag, Berlin, etc., 1979. [Русский перевод: Айгнер М. *Комбинаторная теория.* — М.: Мир, 1982.]

[BaKT99] Barg A., E. Korzhik, H.C.A. van Tilborg, *On the complexity of minimum distance decoding of long linear codes*, to appear in the IEEE Transactions on Information Theory.

[Baue97] Bauer F.L. *Decrypted Secrets; Methods and Maxims of Cryptology*, Springer Verlag, Berlin, etc., 1997. [Готовится русский перевод: Бауэр Ф. *Расшифрованные секреты. Методы и принципы криптологии.*. — М.: Мир, 2004(план).]

[BekP82] Beker H., F. Piper, *Cipher Systems*, the Protection of Communications, Northwood Books, London, 1982.

[Berl68] Berlekamp E.R. *Algebraic Coding Theory*, McGraw-Hill Book Company, New-York, etc., 1968. [Русский перевод: Берлекэмп Э., *Алгебраическая теория кодирования.* — М.: Мир, 1971.]

[BeMT78] Berlekamp E.R., R.J. McEliece, H.C.A. van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Transactions on Information Theory, IT–24, pp. 384–386, 1978.

[BeJL86] Beth T., D. JungNickel, H. Lenz, *Design Theory*, Cambridge University Press, Cambridge, etc., 1986.

[BihS93] Biham E., A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer Verlag, New-York, etc., 1993.

[BoDML97] Boneh D., R.A. DeMillo, R.J. Lipton, *On the importance of checking cryptografic protocols for faults*, Advances in Cryptology: Proc. of Eurocrypt'97, W. Fumy, Ed., Lecture Notes in Computer Science 1233, Springer Verlag, Berlin, etc., pp. 37–51, 1997.

[Bos92] Bos J.N.E. *Practical privacy*, Ph.D. Thesis, Eindhoven University of Technology, the Netherlands, 1992.

[Bric85] Brickell E.F. *Breaking iterated knapsacks*, in Advances in Cryptography: Proc. of Crypto'84, G.R. Blakley and D. Chaum, Eds., Lecture Notes in Computer Science 196, Springer Verlag, Berlin, etc., pp. 342–358, 1985.

[Bric89] Brickell E.F. *Some ideal secret sharing schemes*, The Journal of Combinatorial Mathematics and Combinatorial Computing, Vol. 6, pp. 105–113, 1989.

[CanS98] Canteaut A., N. Sendrier, *Cryptanalysis of the original McEliece cryptosystem*, Advances in Cryptology: Proc. AsiaCrypt'98, K. Ohta and D. Pei, Eds., Lecture Notes in Computer Science 1514, Springer Verlag, Berlin, etc., pp. 187–199, 1998.

[ChoR85] Chor B., R.L. Rivest, *A knapsack type public key cryptosystem based on arithmetic in finite fields*, in Advances in Cryptography: Proc. of Crypto'84, G.R. Blakley and D. Chaum, Eds., Lecture Notes in Computer Science 196, Springer Verlag, Berlin, etc., pp. 54–65, 1985.

[CohL82] Cohen H., H.W. Lenstra Jr. *Primality testing and Jacobi sums*, Report 82–18, Math. Inst., Univ. of Amsterdam, Oct. 1982. [Русский перевод: Коэн Х., Ленстра Х. (мл.) *Проверка чисел на простоту и суммы Якоби*// Кибернетический сборник, вып. 24. — М.: Мир, 1987, сс. 101–146.]

[Cohn77] Cohn P.M. *Algebra*, Vol. 2, John Wiley & Sons, London, etc., 1977.

[Copp84] Coppersmith D. *Fast evaluation of logarithms in fields of characteristic two*, IEEE Transactions on Information Theory, IT–30, pp. 587 594, 1984.

[CopFPR96] Coppersmith D., M. Franklin, J. Patarin, M. Reiter, *Low-exponent RSA with Related Messages*, Advances in Cryptology: Proc. of Eurocrypt'96, U. Maurer, Ed., Lecture Notes in Computer Science 1070, Springer Verlag, Berlin, etc., pp. 1–9, 1996.

[CovM67] Coveyou R.R., R.D. McPherson, *Fourier analysis of uniform random number generators*, J. Assoc. Comput. Mach., 14, pp. 100–119, 1967.

[Demy94] Demytko N. *A new elliptic curve based analogue of RSA*, Advances in Cryptology: Proc. of Eurocrypt'93, T. Helleseth, Ed., Lecture Notes in Computer Science 765, Springer Verlag, Berlin, etc., pp. 40–49, 1994.

[Denn82] Denning D.E.R. *Cryptography and Data Security*, Addison-Wesley publ., Comp., Reading Ma, etc., 1982.

[DifH76] Diffie W., M.E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, IT–22, pp. 644- 654, 1976.

[Dijk97] Dijk M. van, *Secret Key Sharing and Secret Key Generation*, Ph. D. Thesis, Eindhoven University of Technology, the Netherlands, 1997.

[ElGa85] ElGamal T. *A public–key cryptosystem and a signature scheme on discrete logarithms*, Advances on Cryptology: Proc of Crypto'84, G.R. Blakley and D. Chaum, Eds., Lecture Notes in Computer Science 196, Springer Verlag, Berlin, etc., pp. 10–18, 1985.

[FiaS87] Fiat A., A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, Advances in Cryptology: Proc. of Crypto'86. A.M. Odlyzko, Ed., Lecture Notes in Computer Science 263, Springer Verlag, Berlin, etc., pp. 186–194, 1987.

[FIPS94] FIPS 186, *Digital Signature Standard*, Federal Information Processing Standards Publication 186, U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia, 1994.

[Frie73] Friedman W.F. *Cryptology*, in Encyclopedia Britanica, p. 848, 1973.

[GarJ79] Garey M.R., D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, W.H. Freeman and Co., San Fransisco, 1979. [Русский перевод: Гэри М., Джонсон Д., *Вычислительные машины и трудно решаемые задачи*, — М.: Мир, 1982.]

[GilMS74] Gilbert E.N., F.J. MacWilliains, N.J.A. Sloane, *Codes which detect deception*, Bell System Technical Journal, Vol. 53, pp. 405–424, 1974.

[Golo67] Golomb S.W. *Shift Register Sequences*, Holden-Day, San Fransisco, 1967.

[Hall67] Hall M., Jr. *Combinatorial Theory*, Blaisdell Publishing Company, Waltharn, Ma., 1967. [Русский перевод: Холл М., *Комбинаторика*. — М.: Мир, 1970.]

[HarW45] Hardy G.H., E.M. Wright, *An Introduction to the Theory of Numbers*, Clarendon Press, Oxford, 1945.

[Håst88] Håstad J. *Solving simultaneous modular equations of low degree*, SIAM Journal on Computing, 17, pp. 336-341, 1988.

[HelR83] Hellman M.E., J.M. Reyneri, *Fast computation of discrete logarithms over GF(q)*, in Advances in Cryptography: Proc. of Crypto'82, D. Chaum, R. Rivest, A. Sherman, Eds., Plenum Publ. Comp., New York, pp. 3–13, 1983.

[Huff52] Huffman D.A. *A method for the costruction of minimum-redundancy codes*, Proc. IRE, 14, pp. 1098–1101, 1952.

[Joha94a] Johansson T. *A shift register of unconditionally secure authentication codes*, Designs, Codes and Cryptograpy, 4, pp. 69–81, 1994.

[Joha94b] Johansson T. *Contributions to Uncoditionally Secure Authentication*, KF Sigma, Lund, 1994.

[JohKS93] Johansson T., G. Kabatianskii, B. Smeets, *On the relation between A-codes and codes correcting independent errors*, Advances in Cryptography: Proc. of Eurocrypt'93, T. Helleseth, Ed., Lecture Notes in Computer Science 765, Springer Verlag, Berlin, etc., pp. 1–10, 1993.

[Kahn67] Kahn D. *The Codebreakers, the Story of Secret Writing*, Macmillan Company, New York, 1967. [Русский перевод: Кан Д., *Взломщики кодов*. — М.: Центрполиграф, 2000.]

[Khin57] Khinchin A.I. *Mathematical Foundations of Information Theory*, Dover Publications, New York, 1957.

[Knud94] Knudsen L.R. *Block Ciphers — Analysis, Designs and Applications*, PhD Thesis, Computer Science Department, Aarhus University, Denmark, 1994.

[Knut69] Knuth D.E. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, Addison-Wesley, Reading, MA., 1969. [Русский перевод: Кнут Д. *Искусство программирования для ЭВМ, т. 2. Получисленные алгоритмы.* — М.: Мир, 1977.]

[Knut73] Knuth D.E. *The Art of Computer Programming, Vol. 3, Sorting and searching*, Addison-Wesley, Reading, MA., 1973. [Русский перевод: Кнут Д. *Искусство программирования для ЭВМ, т. 3. Сортировка и поиск.* — М.: Мир, 1978.]

[Knut81] Knuth D.E. The Art of Computer Programming, Vol. 2, Seminumerical Algorithms, Addison-Wesley, Reading, MA., 1981. [Русский перевод: Кнут Д. *Искусство программирования, т. 2. Получисленные методы*, — С.-П.: Вильяме, 2000.]

[Koch96] Kocher P.C. *Timing attacks on implementations of Diffie – Hellman, RSA, DSS, and Other Systems*, Advances in Cryptology: Proc. of Crypto'96, N. Koblitz, Ed., Lecture Notes in Computer Science 1109, Springer Verlag, Berlin, etc., pp. 104–113, 1996.

[Konh81] Konheim A.G. *Cryptography, a Primer*, John Wiley & Sons, New York, etc., 1981.

[Kraf49] Kraft L.G. *A Device for Quantizing, Grouping and Coding Amplitude Modulated Pulses*, MS Th., Dept. of EE, MIT, Cambr., Mass., 1949.

[LagO83] Lagarias J.C., A.M. Odlyzko, *Solving low-density subset problems*, Proc. 24th Annual IEEE Symp. on Found, of Comp. Science, pp. 1-10, 1983.

[Lai92] Lai X. *On the design and security of block ciphers*, ETH Series in Information Processing, J.J. Massey, Ed., Vol. 1, Hartung-Gorre Verlag, Konstanz, 1992.

[LeeB88] Lee P.J., E.F. Brickell, *An observation on the security of McEliece's public-key cryptosystem*, in Advances in Cryptography: Proc. of Eurocrypt'88, C.G. Gunther, Ed., Lecture Notes in Computer Science 330, Springer Verlag, Berlin, etc., pp. 275–280, 1988.

[Lehm76] Lehmer D.H. *Strong Carmichael numbers*, J. Austral. Math. Soc., Ser. A 21, pp. 508–510, 1976.

[LensA96] Lenstra A.K. *Memo on RSA signature generation in the presence of faults*, Sept. 1996.

[LenLL82] Lenstra A.K., H.W. Lenstra, L. Lovász, *Factoring polynomials with rational coefficients*, Math. Annalen, 261, pp. 515–534, 1982.

[LensH83] Lenstra H.W. Jr. *Fast prime number tests*, Nieuw Archiev voor Wiskunde (4) 1, pp. 133–144, 1983.

[LensH86] Lenstra H.W. Jr. *Factoring integers with elliptic curves*, Report 86–16, Dept. of Mathematics, University of Amsterdam, Amsterdam, the Netherlands.

[Liu68] Liu C.L. *Introduction to combinatorial mathematics*, McGraw-Hill, New York, 1968.

[Lüne87] Lüneberg H. *On the Rational Normal Form of Endomorphisms; a Primer to Constructive Algebra*, BI Wissenchaftsverlag, Mannheim, etc., 1987.

[MacWS77] MacWilliams F.J., N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland Publ. Comp., Amsterdam, etc., 1977. [Русский перевод: Мак-Вильяме Ф.Дж., Слоэн Н.Дж.А. *Теория кодов, исправляющих ошибки.* — М.: Связь, 1979.]

[Mass69] Massey J.L. *Shift-register synthesis and BCH decoding*, IEEE Transactions on Information Theory, IT–15, pp. 122–127, 1969.

[MatY93] Matsui M., A. Yamagishi, *A new method for known plaintext attack of FEAL cipher*, Advances in Cryptology: Proc. Eurocrypt'92, R.A. Rueppel, Ed., Lecture Notes in Computer Science 658, Springer Verlag, Berlin, etc., pp. 81–91, 1993.

[Maur92] Maurer U. *A universal statistical test for random bit generators*, Journal of Cryptology, 5, pp. 89–105, 1992.

[McEl78] McEliece R.J. *A public-key cryptosystem based on algebraic coding theory*, JPL DSN Progress Report 42–44, pp. 114–116, 1978.

[McEl81] McEliece R.J., D.V. Sarwate, *On sharing secrets and Reed - Solomon codes*, Comm. ACM, Vol 24, pp. 583–584, 1981.

[McMi56] McMillan B. *Two inequalities implied by unique decipherability*, IEEE Transactions on Information Theory, IT–56, pp. 115–116, 1956.

[Mene93] Menezes A.J. *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston, etc., MA, 1993.

[MeOkV93] Menezes A.J., T. Okamoto, S.A. Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, IEEE Transactions on Information Theory, IT–39, pp. 1639–1646, 1993.

[MeOoV97] Menezes A.J., P.C. van Oorshot, S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, etc., 1997.

[MerH78] Merkle R.C., M.E. Hellman, *Hiding information and signatures in trapdoor knapsacks*, IEEE Transactions on Information Theory, IT–24, pp. 525–530, 1978.

[MeyM82] Meyer C.H., S.M. Matyas, *Cryptography: a New Dimension in Computer Data Security*, John Wiley & Sons, New York, etc., 1982.

[Mill76] Miller G.L. *Riemann's hypothesis and tests for primality*, Journal of Computer and System Sciences, 13, pp. 300–317, 1976. [Русский перевод: Миллер Г.Л. *Гипотеза Римана и способы проверки простоты чисел*//Кибернетический сб., вып. 23. — М.: Мир, 1986, с. 31-50.]

[Mill86] Miller G.L. *Use of elliptic curves in cryptography*, Advances in Cryptology: Proc. Crypto'85, H.C. Williams, Ed., Lecture Notes in Computer Science 218, Springer Verlag, Berlin, etc., pp. 417 426, 1986.

[Moni80] Monier L. *Evaluation and comparison of two efficient probabi-*

listic primality testing algorithms, Theoretical Computer Science, 12, pp. 97–108, 1980.

[MorB75] Morrison M.A., J. Brillhart, *A method of factoring and the factorization of F_7*, Math. Comp. 29, pp. 183–205, 1975.

[Nied86] Niederreiter H. *Knapsack type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory, 15, pp. 159–166, 1986.

[NybR93] Nyberg K., R.A.Rueppel, *A new signature scheme based on the DSA giving message recovery*, 1st ACM Conference on Computer and Communications Security, ACM Press, pp. 58–61, 1993.

[Odly85] Odlyzko A.M. *Discrete logarithms in finite fields and their cryptographic significance*, Advances in Cryptology: Proc. Eurocrypt'84, T. Beth, N. Cot and I. Ingemarsson, Eds., Lecture Notes in Computer Science 209, Springer Verlag, Berlin, etc., pp. 224–314, 1985.

[Patt75] Patterson N.J. *The algebraic decoding of Goppa codes*, IEEE Transactions on Information Theory, IT–21, pp. 203–207, 1975.

[Pera86] Peralta R. *A simple and fast probabilistic algorithm for computing square roots modulo a prime number*, presented at Eurocrypt'86, J.L. Massey, Ed., no proceedings published.

[PohH78] Pohlig S.C., M.E. Hellman, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Transactions on Information Theory, IT–24, pp. 106–110, 1978.

[Poll75] Pollard J.M. *A Monte Carlo method for factoring*, BIT–15, pp. 331–334, 1975.

[Poll78] Pollard J.M. *Monte Carlo methods for index computations* (mod p), Mathematics of Computations, 32, pp. 918–924, 1978.

[Rabi79] Rabin M.O. *Digitalized signatures and public-key functions as intractable as factorization*, MIT/LCS/TR–212, MIT Lab. for Comp. Science, Cambridge, Mass., 1979.

[Rabi80a] Rabin M.O. *Probabilistic algorithms for testing primality*, Journal of Number Theory, 12, pp. 128–138, 1980.

[Rabi80b] Rabin M.O. *Probabilistic algorithms in finite fields*, SIAM J. Comput, 80, pp. 273–280, 1980.

[RisL79] Rissanen J., G. Langdon, *Arithmetic coding*, IBM Journal of Research and Development, 23, pp. 149–162, 1979.

[RivSA78] Rivest R.L., A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM, Vol 21, pp. 120–126, 1978.

[Rose84] Rosen K.H. *Elementary Number Theory*, Addison-Wesley Publ. Comp., Reading, Mass., 1984.

[Ruep86] Rueppel R.A. *Analysis and Design of Streamciphers*, Springer Verlag, Berlin, etc., 1986,

[SatA98] Satoh T., K. Araki, *Ferma quotients and the polynomial time*

discrete log algorithm for anomalous elliptic curves, Commentarii Mathematici Universitatis Sancti Pauli, 47, pp. 81–92, 1998.

[Schne96] Schneier B. *Applied Cryptography, 2nd Edition*, John Wiley & Sons, New York, etc., 1996.

[Schno90] Schnorr C.P. *Efficient identification and signatures for smart cards*, Advances in Cryptology, Crypto'89, G. Brassard, Ed., Lecture Notes in Computer Science 435, Springer Verlag, Berlin, etc., pp. 239–252, 1990.

[Schno91] Schnorr C.P. *Efficient signature generation by smart cards*, Journal of Cryptology, 4, pp. 161–174, 1991.

[Scho95] Schoof R. *Counting points on elliptic curves over finite fields*, Journal de Theorie des Nombres de Bordeaux, 7, pp. 219–254, 1995.

[Sham79] Shamir A. *How to share a secret*, Communications of the ACM, Vol. 22, pp. 612–613, 1979.

[Sham82] Shamir A. *A polynomial time algorithm for breaking the basis Merkle–Hellman cryptosystem*, in Proc. 23-rd IEEE Symp. Found. Computer Science, pp. 145–152, 1982.

[Shan49] Shannon C.E. *Communication theory and secrecy systems*, BSTJ, 28, pp. 656–715, 1949. [Русский перевод: Шеннон К. *Теория связи в секретных системах*//Шеннон К., Работы по теории информации и кибернетике. — М.: ИЛ, 1963, с. 333–369.]

[Shap83] Shapiro H.N. *Introduction to the Theory of Numbers*, John Wiley & Sons, New York, etc., 1983.

[Silv86] Silverman J.H. *The Arithmetic of Elliptic Curves*, Springer Verlag, Berlin, etc., 1986.

[Silv98] Silverman J.H. *The XEDNI calculus and the elliptic curve discrete logarithm problem*, preprint.

[SilT92] Silverman J.H., J. Tate, *Rational Points on Elliptic Curves*, Undergraduate Texts in Mathematics, Springer Verlag, New York, etc., 1992.

[Simm92] Simmons G.J. *A survey of information authentication*, in Contemporary Cryptology: the Science of Information Integrity, G.J. Simmons, Ed., IEEE Press, New York, pp. 379–419, 1992.

[Smar98] Smart N. *The discrete logarithm problem on elliptic curves of trace one*, Journal of Cryptology, to appear.

[SolS77] Solovay R., V. Strassen, *A fast Monte-Carlo test for primality*, SIAM J. Comput., 6, pp. 84–85, 1977.

[Stin95] Stinson D.R. *Cryptography: Theory and Practice*, CRC Press, Inc., Boca Raton, 1995.

[SugK76] Sugiyama Y., M. Kashara, S. Hirasawa, T. Namekawa, *An erasures-and-errors decoding algorithm for Goppa codes*, IEEE Transactions on Information Theory, IT–22, pp. 238–241, 1976.

[vTbu88] van Tilburg H. *On the McEliece public-key cryptosystem*, Advances in Cryptography: Proc. of Crypto'88, S. Goldwasser, Ed., Lecture Notes in Computer Science 403, Springer Verlag, Berlin, etc., pp. 119–131, 1989.

[Vaud98] Vaudenay S. *Cryptanalysis of the Chor – Rivest cryptosystem*, Advances in Cryptography: Proc. of Crypto'98, H. Krawczyk, Ed., Lecture Notes in Computer Science 1462, Springer Verlag, Berlin, etc., pp. 243–256, 1998.

[VerT97] Verheul E.L., H.C.A. van Tilborg, *Constructions and properties of k out of n visual secret sharing schemes*, Designs, Codes and Cryptography, Vol. 11, No. 2, pp. 179–196, 1997.

[Well99] Wells R.B. *Applied Coding and Information Theory*, Prentice Hall, Upper Saddle River NJ, 1999.

[Wien90] Wiener M.J. *Cryptanalysis of short RSA secret exponents*, IEEE Transactions on Information Theory, IT–36, pp. 553–558, 1990.

[ZivL77] Ziv J., A. Lempel, *A universal algorithm for sequential data compression*, IEEE Transactions on Information Theory, IT–23, pp. 337–343, 1977.

[ZivL78] Ziv J., A. Lempel, *Compression of individual sequences by variable rate coding*, IEEE Transactions on Information Theory, IT 24, pp. 530–536, 1978.

Список литературы, добавленной при переводе

*[АйеР87] Айерлэнд К., Роузен М. *Классическое введение в современную теорию чисел.* — М.: Мир, 1987.

*[АЗКЧ01] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. *Основы криптографии.* — М.: Гелиос АРВ, 2001.

*[Вас03] Василенко О.Н. *Теоретико-числовые алгоритмы в криптографии.* — М.: МЦНМО, 2003.

*[ГОСТ01] ГОСТ Р 34.10-2001. *Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи.* — М.: препринт, 2001.

*[ГОСТ96] ГОСТ 28147-89. *Защита криптографическая. Алгоритм криптографического преобразования.* — М.: Изд-во стандартов, 1996.

*[ГОСТП94] ГОСТ Р 34.10-94. *Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма.* — М.: Изд-во стандартов, 1994.

*[ГОСТХ94] ГОСТ Р 34.11-94. *Криптографическая защита информации. Функция хэширования.* — М.: Изд-во стандартов, 1994.

*[Кобл01] Коблиц Н. *Курс теории чисел и криптографии.* — М.: ТВП, 2001.

*[Ленг70] Ленг С. *Введение в теорию диофантовых приближений.* — М.: Мир, 1970.

*[Ленс91] Ленстра Х.В. (мл.), *Эллиптические кривые и теоретико-числовые алгоритмы*//Международный конгресс математиков в Беркли. — М.: Мир, 1991, с. 164–193.

*[ЛидН88] Лидл Р., Нидеррайтер Г. *Конечные поля*, в 2 тт. — М.: Мир, 1988.

*[ЛидП96] Лидл Р., Пильц Г. *Прикладная абстрактная алгебра.* — Екатеринбург: Изд-во УрГУ, 1996.

*[Молд98] Молдовян Н.А. *Скоростные блочные шифры.* — СПб.: Изд-во СПбГУ, 1998.

*[Неча99] Нечаев В.И. *Элементы криптографии (основы теории защиты информации).* — М.: Высшая школа, 1999.

*[НодК99] Ноден П., Китте К. *Алгебраическая алгоритмика.* — М.: Мир, 1999.

*[Прах67] Прахар К. *Распределение простых чисел.* — М.: Мир, 1967.

*[РосМ01] Ростовцев А.Г., Михайлова Н.В. *Методы криптоанализа классическмх шифров.* — 2001 (препринт).

*[Сало96] Саломаа А. *Криптография с открытым ключом.* — М.: Мир, 1996.

*[Серр72] Серр Ж.-П. *Курс арифметики.* — М.: Мир, 1972.

*[Слоэ83] Слоэн Н.Дж.А. *Коды, исправляющие ошибки, и криптография.*/ / В сб. "Математический цветник". — М.: Мир, 1983, с. 432–472.

*[Хинч78] Хинчин А.Я. *Цепные дроби.* — М.: Наука, 1978.

*[Холл62] Холл М. *Теория групп.* — М.: ИЛ, 1962.

*[Чанд74] Чандрасекхаран К. *Введение в аналитическую теорию чисел.* — М.: Мир, 1974.

*[Чер02] Черемушкин А.В. *Лекции по арифметическим алгоритмам в криптографии.* — М.: МЦНМО, 2002.

*[Ящен98] Ященко В.В. (ред.), *Введение в криптографию.* — М.: МЦНМО, 1998.

*[AgKS02] Agrawal M., Kayal N., Saxena N. *Primes is in P*, Kanpur, India, 2002 (preprint).

*[Bone99] Boneh D. *Twenty years of attacks on the RSA cryptosystem*, Notices of AMS, Vol. 46, No. 2, 1999, pp. 203–213.

*[DaRi99] Daemen J., Rijmen V. *AES proposal: RijnDael*, 1999, preprint.

*[LandOO] Landau S. *Stanning the test of time: the Data Encryption Standard*, Notices of AMS, Vol. 47, No. 3, 2000, pp. 341–349.

*[KaPS95] Kaufman C., R. Perlman, M. Speciner, *Network Security.* Private communication in a public world, Prentice Hall, New Jersey, 1995.

Символы и обозначения [1]

(a, b)	наибольший общий делитель, А.1, опр. А.2	336
$[a, b]$	наименьшее общее кратное, А.1, опр. А.3	337
$\left(\frac{u}{m}\right)$	символ Якоби, А.4, опр. А.11	354
R/S	кольцо классов вычетов, В.1.1, теор. В.2	379
$(s(x))$	идеал, порожденный $s(x)$, В.2	389
$\equiv$	сравнение, А.3.1, опр. А.4	343
$\|\|v\|\|$	длина вектора, В.1.2	384
$U^{\perp}$	ортогональное дополнение, В.1.2, опр. В.13	385
$\Gamma(p_U(x), \mathrm{GF}(2^m))$	код Гоппы, 11.1, теор. 11.1	230
μ	функция Мёбиуса, А.6.1, опр. А.14	368
$\pi(x)$	число простых$\leq x$, А.1, опр. А.1	336
φ	функция Эйлера, А.3.2, опр. А.6	345
χ	символ Лежандра, А.4, опр. А.10	354
$\Omega(f)$	выходное пространство LFSR, 3.2.2, опр. 3.4	43
$AC(k)$	автокорреляция, 3.1, (3.1)	36
D_n	избыточность, 5.1, опр. 5.1	83
$d(u)$	плотность рюкзака, 12.2.1, опр. 12.2	261
$\mathcal{E}$	(1) традиционная криптосистема, 1.2, опр. 1.1	12
	(2) эллиптическая кривая, 10.1, опр. 10.1	209
f^*	взаимный многочлен, 3.2.2, (3.6)	43
$\mathrm{F}[x]$	кольцо многочленов над F, В.2	386
$\mathbb{F}_q$	конечное поле из q элементов, В.1.1	377
GF	поле Галуа, В.1.1	377
$h(p)$	энтропия, 5.1, (5.4)	72
$H(X)$	энтропия, 5.1	72
$H(X\|Y)$	условная энтропия, 5.2, (5.7)	86
$I_q(n)$	число неприводимых многочленов степени n над $\mathbb{F}_q$, В.3, опр. В.15	391
$I(n)$	число бинарных неприводимых многочленов степени n, В.3, опр. В.15	391
$I(X, Y)$	взаимная информация, 5.2, (5.8)	87
L_k	линейная сложность, 3.3.2, опр. 3.6	59
$\mathcal{N}$	непривилегированное множество (структуры доступа), 15.1, опр. 15.1	315
NQR	квадратичный невычет, А.4, опр. А.9	354
НОД	наибольший общий делитель, А.1, опр. А.2	336
НОК	наименьшее общее кратное, А.1, опр. А.3	337
P_D	вероятность успешного обмана, 13.3.1, (13.4)	285

[1] Мы максимально локализуем каждое обозначение. — *Прим. ред.*

P_I	вероятность успешной атаки имперсонификации, 13.3.1, опр. 13.3	285
P_S	вероятность успешной атаки подмены, 13.3.1, опр. 13.4	285
$\mathcal{P}$	привилегированное множество (системы доступа), 15.1, опр. 15.1	315
$Q^{(d)}$	многочлен деления круга, B.4.6, опр. B.19	410
QR	квадратичный вычет, A.4, опр. A.9	354
Tr	функция следа, B.5, зад. B.16	414
$\mathrm{V}(n,q)$	n-мерное векторное пространство над $\mathrm{GF}(q)$, 13.3.4	301
$w(x)$	вес вектора, 11.2.1	235
$\mathbb{Z}_p$	целые числа по модулю p, B.2	386

Предметный указатель

Оглавление

Учебное издание

Хенк К.А. ван Тилборг

ОСНОВЫ КРИПТОЛОГИИ
Профессиональное руководство и интерактивный учебник

Зав. редакцией академик *В. И. Арнольд*
Зам. зав. редакцией *А. С. Попов*
Ведущий редактор *М. С. Стригунова*
Художник *М. М. Иванов*
Технический редактор *Е. В. Денюкова*

Оригинал макет подготовлен *Д. С. Ананичевым* в пакете LaTeX 2_ε
с использованием кириллических шрифтов семейства LH.

Подписано к печати 21.10.2005. Формат $70 \times 100^1/_{16}$.
Печать офсетная. Объем 14,75 бум.л. Усл. печ. л. 38,35
Изд. №1/9938. Тираж 1500 экз. Заказ 12009

Издательство "Мир"
Министерство культуры и массовых коммуникаций РФ.
107996, ГСП–6. Москва, 1-й Рижский пер., д. 2

Отпечатано с готовых оригинал-макетов
в ОАО «ИПК «Южный Урал».
460000, г. Оренбург, пер. Свободина, 4.

Х.К.А. ван Тилборг

ОСНОВЫ КРИПТОЛОГИИ

ПРОФЕССИОНАЛЬНОЕ РУКОВОДСТВО И ИНТЕРАКТИВНЫЙ УЧЕБНИК

«ОСНОВЫ КРИПТОЛОГИИ» – перевод обновленной и улучшенной версии «Введения в криптологию», опубликованного в оригинале в 1988 г. Помимо ревизии имевшегося материала добавлено много новых разделов и две новые главы – об эллиптических кривых и кодах аутентификации. Кроме этого, книга сопровождается полной электронной версией текста на CD-ROM'е в качестве интерактивного учебника в пакете «Mathematica».

«ОСНОВЫ КРИПТОЛОГИИ» будут интересны специалистам в информатике – исследователям и практикам, а также математикам и студентам, изучающим криптологию.

Включает интерактивный учебник на CD-ROM'е

ISBN 5-03-003639-3