КРИПТОГРАФИЯ С ОТКРЫТЫМ КЛЮЧОМ

Arto Salomaa

Public-Key Cryptography

With 18 Figures

Springer-Verlag
Berlin Heidelberg New York London
Paris Tokyo Hong Kong Barcelona

А. Саломаа

Криптография с открытым ключом

Перевод с английского
И.А. Вихлянцева
под редакцией
А.Е. Андреева и А.А. Болотова

Москва "Мир"
1995

ББК 22.1
С20
УДК 51(0.062)

Саломаа А.

С20 Криптография с открытым ключом: Пер. с англ. — М.: Мир, 1995. — 318 с., ил.

ISBN 5–03–001991–X

Предлагаемая вниманию читателя книга является практически первым изданием на русском языке, посвященным математическим вопросам криптографии, и написана известным финским ученым, автором нескольких монографий и учебников А. Саломаа. В ней обобщаются последние достижения в области криптографии, особое внимание при этом уделяется криптосистемам с открытым ключом. Также рассматриваются криптографические протоколы, которые открывают новую страницу в обеспечении защиты информационных потоков в различных компьютерных сетях.

Для всех интересующихся проблемами криптографии и защитой информации.

ББК 22.1

Редакция литературы по математическим наукам

ISBN 5–03–001991–X(русск.)
ISBN 0–387–52831–8(англ.)

От редакторов перевода

В настоящее время криптография переживает бум. На протяжении своей более чем тысячелетней истории она обслуживала в основном нужды военных, дипломатов и разведчиков в обеспечении секретности при обмене сообщениями и представляла собой постоянно обновляющийся и совершенствующийся набор технических приемов шифрования (и соответственно дешифрования), которые, естественно, сами держались в секрете. И лишь с середины семидесятых годов (в связи с изобретением систем с открытым ключом) криптография не только перестала быть секретным сводом приемов шифрования/дешифрования, но и начала оформляться в новую математическую теорию и стала объектом интенсивного математического изучения.

Предлагаемая вниманию читателя книга — практически первая на русском языке, посвященная целиком математическим вопросам криптографии, — обобщает последние (до 1990 г.) достижения в области криптографии и написана известным ученым, действительным членом Финской академии наук, почетным доктором нескольких европейских университетов, профессором Университета Турку Арто Саломаа. Он является автором нескольких превосходных монографий и учебников, переведенных на многие языки мира, в том числе и на русский (А. Саломаа. Жемчужины теории формальных языков, — М.: Мир, 1986, 180 с.) и свыше ста статей.

В первой главе дается обзор классических криптосистем — от системы Цезаря до криптосистемы DES (стандарт шифрования данных) — с многочисленными и весьма полезными примерами. Основное содержание книги посвящено криптосистемам с открытым ключом (гл. 2–5). В гл. 6 рассматриваются криптографические протоколы, возможно, наиболее важные в прикладном отношении вопросы в криптографии, открывающие новую страницу в обеспечении защиты информационных потоков в различных компьютерных сетях.

Представление этого материала с методической точки зрения безупречно, текст книги содержит много примеров и может послужить хорошим учебником для начинающих изучать эту науку. Изложение ведется строго, но без излишнего формализма. Специалисты по криптографии найдут для себя некоторые новые результаты.

Надеюсь, что книга привлечет внимание широкой математической общественности к новой, бурно развивающейся за рубежом математической дисциплине и станет доступной для отечественных специалистов по математическим методам обеспечения безопасности и конфиденциальности компьютерного общения и коммуникаций в компьютерных сетях (в частности, защита информационных потоков в банковских сетях,

в системах голосования при организации выборов с использованием компьютеров и т.д.).

Главы 1, 2, 4, 5 переведены на русский язык И. А. Вихлянцевым, гл. 3, 6 и приложения — А. А. Болотовым. Компьютерные верстка и набор в системе LaTeX осуществлены И. А. Вихлянцевым.

В заключение пользуюсь возможностью выразить глубокую благодарность и искреннюю признательность автору, проявившему бескорыстный интерес и внимание к русскому изданию своей книги.

А. А. Болотов, А. Е. Андреев

Памяти моей сестры
Сиркки Саломаа 1919–1989

Предисловие

Криптография, или тайнопись, вероятно, так же стара, как и письменность в целом. Но лишь в последнее время она стала объектом широкомасштабных научных исследований. Одной из причин этого послужило большое число новых приложений в области защиты информации. Возможно, даже более важной причиной огромного роста научных исследований в криптографии явилась плодотворная идея криптографии с открытым ключом, создавшая новые перспективы и возможности в обмене информацией.

Эта книга представляет собой взгляд на криптографию с открытым ключом с позиций классической криптографии. В книге была сделана попытка представить некоторые из самых современных достижений и результатов. Приводимые в примерах для шифрования тексты составляют пакет основных сведений о сауне.

Благодарности. Hermann Maurer в конце семидесятых годов пробудил мой интерес к криптографии. С 1983 г. я использовал несколько версий этой книги в курсах криптографии в Университетах Турку и Лейдена, а также в Техническом университете Вены. Замечания слушателей этих курсов были полезными. Juha Honkala, Jarkko Kari, Valtteri Niemi, Lila Sântean, Mika Niemi и Ari Renvall критически прочитали различные части рукописи, а первые четверо внесли также свой вклад в многочисленные обсуждения. Для меня также полезными были обсуждения с Ron Book, Wilfried Brauer, Karel Culik, Ferenc Gécseg, Jozef Gruska, Tero Harju, Iiro Honkala, Helmut Jürgensen, Juhani Karhumäki, Werner Kuich, Hannu Nurmi, Kaisa Nyberg, Azaria Paz, Grzegorz Rozenberg, Kai Salomaa, Aimo Tietäväinen, Emo Welzl, Derick Wood и Sheng Yu. Особой благодарности заслуживает Elisa Mikkola за отличный набор рукописи, а также за помощь в различных практических вопросах. Anu Heinimäki нарисовала рисунки. Академия наук Финляндии предоставила мне благоприяные условия для работы. Полезное сотрудничество с сотрудниками академии, особенно с Marjatta

Näätänen, заслуживает искренней признательности. Научная организация MATINE поддержала мои исследования в криптографии. Наконец, я хотел бы поблагодарить издательство Springer-Verlag и особенно доктора Hans Wössner и Mrs. Ingeborg Mayer за хорошее взаимодействие и оперативность издания книги.

Турку, Май 1990 Арто Саломаа

Глава 1

Классическая криптография

1.1. Криптосистемы и криптоанализ

Наука и искусство криптографии имеют две стороны. Существует мир легальных коммуникаций, частью которого являются, к примеру, легальные пользователи банковских и биржевых сообщений. Этот мир можно рассматривать как открытый и залитый солнечным светом. Однако существует и темный мир врага, прибегающего ко всевозможным запрещенным приемам. И если люди легального мира хотят, чтобы враг как можно меньше понимал содержание их сообщений, то врагу, напротив, нравится иметь дело с посланиями, которые легко расшифровать.

Криптография — это продолжение борьбы между двумя мирами. Необходимость разработки более сильных методов в легальном мире продиктована успехами врагов и означает в свою очередь новый вызов в их адрес. Таким образом, борьба продолжается.

Как представлены два этих мира в книге? Что касается прошлого, то тут никаких трудностей нет. Сначала описывается метод легального мира и затем объясняется, как враг предпринял успешную атаку. Сегодня же ситуация в корне меняется. Всякий раз, когда описывается успешная вражеская атака, мы должны признать, что соответствующие методы легального мира все же не были надежными. Нельзя достигнуть успеха одновременно в обоих мирах.

Можно действовать следующим образом: сначала в деталях дать

метод для легального мира, затем в общих чертах набросать некоторые возможные варианты вражеских атак, при этом одновременно пояснить, почему эти атаки скорее всего не будут иметь успеха. Это, конечно, не исключает возможного триумфа некоторых других, может быть очень изобретательных, вражеских атак. Так или иначе, далее в книге используется данный подход. Вероятность безопасности методов часто является очень высокой, хотя математически это не всегда можно доказать.

Следующее замечание применимо к обоим мирам. Хотя мы называем их "легальный" и "темный", это не всегда означает, что первые — "хорошие парни", а последние — из Мордора, где живет Саурон[1]. В различных ситуациях роли могут меняться. Например, перехват сообщений может применяться нашей страной во время войны, тогда как посланиями обменивался наш враг. Конечно, справедливость на нашей стороне! Или другой пример: легальные пользователи банка могут быть преступниками, а полиция должна расследовать их действия. Терминология, введенная ниже, будет беспристрастной, т.е. никаких суждений относительно того, какая из противоборствующих сторон права, мы делать не будем.

Теперь мы готовы ввести фундаментальные понятия криптографии. Они будут использоваться на протяжении всей книги. Подчеркнем, что терминология отнюдь не однозначна и фиксирована для различных описаний секретного письма. При введении терминологии мы часто будем приводить и другие обозначения тех же понятий, используемых некоторыми другими авторами.

Наш всеобщий термин для секретного письма — это *криптография*. Он действителен для обоих миров. Некоторые авторы используют термин *криптология* в этом универсальном смысле и резервируют термин *криптография* для деятельности легального мира.

Основная структура изображена на рис. 1.1. Сообщение посылается по незасекреченному каналу, где оно может быть перехвачено с помощью подслушивания.

Картина будет той же самой, говорим ли мы, допустим, о конном курьере или электронной почте. Мы не можем обеспечить безопасность канала, и, следовательно, возможен перехват. Первоочередные цели врага — нарушить секретность связи и извлечь выгоду из тайной информации. Далеко идущие цели могут состоять в следующем. Во-первых, враг может захотеть изменить сообщение, запутав при этом получателя недостоверным текстом, а во-вторых, враг может обмануть получателя, назвав имя ложного отправителя.

[1]Персонаж романа Дж.Р.Р.Толкиена "Властелин колец". — *Прим. перев.*

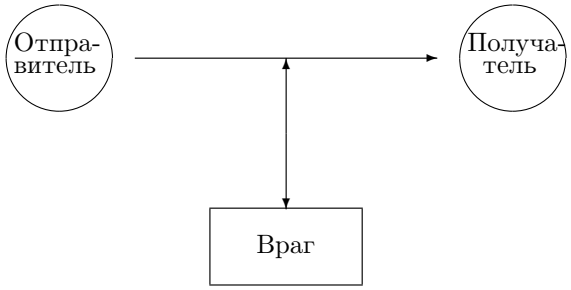


Рис. 1.1.

К примеру, отправитель может послать сообщение “Я не дам поддержки зеленым”. Если враг изменит его на “Я не дам $10.000 для зеленых”, получатель может не догадаться, от кого пришло это совершенно другое по смыслу сообщение.

Враг может также обмануть отправителя, идентифицировав себя в качестве получателя, к примеру, захватив все сообщение и не переслав его законному адресату.

Во всех этих случаях большим преимуществом легальных пользователей было бы то обстоятельство, если бы враг не понимал перехваченного им сообщения. Для этих целей может быть использован некоторый метод шифровки.

Сообщение в его оригинальной форме может быть представлено как *исходный текст*. Затем отправитель *зашифровывает* данный текст. Результатом этого будет *криптотекст*. Теперь криптотекст может быть послан через ненадежный канал. Наконец, получатель *расшифровывает* криптотекст, после чего он/она имеет исходное сообщение.

Таким образом, действия отправителя:

> Зашифрование исходного текста и получение криптотекста.

Действия получателя обратные:

> Расшифрование криптотекста и получение исходного текста.

Мы можем получить также более короткие символьные выражения:

$$E(pt) = ct \qquad \text{и} \qquad D(ct) = pt \ .$$

В литературе часто вместо терминов “исходный текст” и “криптотекст” используется “открытый текст” и “шифртекст”, или кратко “шифр”. Глаголами в этом случае являются “зашифровать” и “расшифровать”. До недавнего времени использовали слово “код” и соответствующие глаголы “кодировать” и “декодировать”, но теперь перестали. Причина этого заключается в том, что слово “код” имеет много других значений: коды, исправляющие ошибки, автоматные коды и т.д. Слово “код” будет ниже использоваться в некоторых специальных случаях, однако не в общем смысле слова “криптотекст”.

Теперь проанализируем зашифрование и расшифрование. Оба этих действия производятся в рамках *криптосистемы*. Криптосистема состоит из следующих компонентов:

1. *Пространство исходных сообщений* PT, которое содержит всевозможные исходные тексты pt.

2. *Ключевое пространство* K. Каждому ключу k в K соответствует алгоритм зашифрования E_k и расшифрования D_k. Если к сообщению pt применить E_k, а к результату шифрования — D_k, то снова получим pt.

3. *Пространство криптотекстов* CT, т.е. набор всевозможных криптотекстов ct. Элементами CT являются результаты применения к элементам PT методов шифрования E_k, где k пробегает все пространство K.

Теперь дадим некоторые основные теоретико-языковые определения. Начнем с конечного непустого множества Σ, называемого *алфавитом*. Элементы алфавита Σ называются *буквами*. Конечные цепочки элементов из Σ называются *словами*. Одна и та же буква может встречаться в слове несколько раз. Слово, содержащее 0 букв, называется *пустым словом* λ. *Длина слова* w — число букв в нем, где каждая буква считается столько раз, сколько раз она появляется. Множество всех слов над Σ обозначим через Σ^*. Подмножества множества Σ^* будем называть *(формальными) языками* над Σ.

Например, если в качестве Σ выбран английский алфавит {A, B, C, . . . , Z}, то ABBA, HORSE и KOKOOKOKOONKOKOKOKKO — слова над Σ (неважно, имеет ли слово какой-нибудь смысл или нет; действительно, третье слово имеет значение лишь на финском). Мы можем также добавить к Σ строчные буквы, все знаки препинания и пустой символ (пробел), необходимые в обычном тексте. Теперь собрание сочинений Шекспира представляет собой слово над этим расширенным алфавитом.

Вернемся к понятию криптосистемы, анализируя далее ее компоненты. Пространство исходных текстов PT обычно состоит из всего множества Σ^* для некоторого алфавита Σ или из всех осмысленных выражений естественного языка. Подчеркнем, что эти две возможности существенно отличаются друг от друга. Если пространством исходных сообщений является Σ^*, то каждая буква в сообщении будет значащей, поэтому нет никакой свободы в процессе расшифрования. С другой стороны, каждый естественный язык имеет высокую избыточность в том смысле, что даже при наличии большого количества ошибок сообщение обычно понимается правильно. Это является огромным преимуществом для перехватчика: он может безошибочно понять сообщение, хотя анализ неверен в нескольких местах! Проиллюстрируем это на примере.

Пример 1.1. Вначале потребуем, чтобы пространство исходных сообщений состояло из английского языка. Рассмотрим сообщение WEMEETTOMORROW (мы игнорируем пробелы между индивидуальными словами; это будет часто делаться и в дальнейшем). Зашифруем его как UBQBBNNFIVPNFOOB (в данный момент мы не говорим, как осуществляется зашифрование). Если анализ перехватчиком криптотекста даст результат WIMIIDTUMAROV, он будет вполне доволен: результат может быть получен верно.

Потребуем теперь, чтобы пространством исходных текстов было все Σ^*, где Σ — двоичный алфавит $\{0,1\}$. Потребуем также, чтобы отправитель и получатель сделали следующее предварительное соглашение, касающееся сообщений: они имеют длину 12 и дают информацию о флоте из 12 кораблей. Посылаемое утром сообщение указывает, какие корабли будут участвовать в действиях текущего дня. К примеру, согласно сообщению 010011000001, участвуют только второй, пятый, шестой и двенадцатый корабли. Сообщения посылаются в зашифрованном виде. Теперь анализ нашего перехватчика должен быть очень аккуратен. Даже если в одном бите будет неточность, может возникнуть серьезная ошибка.

Часто бывает, что исходный текст на английском сперва кодируется в двоичном алфавите, к примеру, заменой каждой буквы двоичным числом, указывающим позицию этой буквы в английском алфавите. Так как $2^4 < 26 < 2^5$, то для этой цели подходят слова длины пять:

$$\mathrm{A} = 00001,\ \mathrm{B} = 00010,\ \mathrm{C} = 00011,\ \ldots,\ \mathrm{N} = 01110,\ \ldots,\ \mathrm{Z} = 11010\ .$$

Для трансляции сообщения без всякой цели маскировки мы будем использовать термины *кодирование* и *декодирование*. В кодировании мы нуждаемся, к примеру, при передаче текстового сообщения. Таким образом, сообщение вначале кодируется, а затем шифруется. Конечно,

с помощью кодирования избыточность естественного языка никак не меняется.

□

После обсуждения пространства исходных сообщений мы дадим несколько комментариев по пространству ключей. Мощность этого пространства не должна быть очень маленькой: перехватчик не должен иметь возможность проверить все ключи. В большинстве случаев пространство ключей бесконечно.

Каждый ключ k, как мы уже говорили, определяет алгоритм зашифрования E_k и алгоритм расшифрования D_k, причем E_k и D_k аннулируют друг друга. Мы не хотим давать более специфических математических характеристик для E_k и D_k, а в действительности даже требовать, чтобы E_k являлся функцией. В некоторых представленных ниже криптосистемах существует много возможностей применения ключа к исходному сообщению, приводящих к различным результатам.

О третьем пункте — пространстве криптотекстов — много говорить не надо. Оно определяется первыми двумя пунктами: все возможные результаты зашифрования всех допустимых исходных сообщений.

Что делает криптосистему хорошей? Сэр Фрэнсис Бекон сформулировал три следующих требования, которые мы представим ниже в нашей терминологии.

1. По заданным E_k и исходному сообщению pt легко вычислить $E_k(pt)$. По заданным D_k и криптотексту ct легко вычислить $D_k(ct)$.

2. Не зная D_k, невозможно восстановить исходное сообщение pt из криптотекста ct.

3. Криптотекст не должен вызывать подозрений, т.е. должен выглядеть естественно.

Можно согласиться с сэром Фрэнсисом, однако имея в виду, что третье требование не может более рассматриваться как важное. В параграфе 1.2 содержится пример, удовлетворяющий этому требованию.

Требование (1) подразумевает, что для легальных пользователей криптосистема не должна быть очень сложной. “Легкость” здесь понимается в рамках теории сложности — см. приложение A. Предполагается, что пользователи имеют приемлемое время вычислений. В требовании (2) “невозможность” заменяется на “трудновычислимость”. Предполагается, что перехватчик также имеет доступ к вычислительной технике. Усиления требования (2) рассматриваются ниже в связи с криптоанализом.

Дополнительные аспекты требования (1) обсуждаются в [Ka]. До появления вычислительной техники при применении какой-либо криптосистемы все делалось вручную. К примеру, армейский генерал, ответственный за криптографию, для проверки новой криптосистемы использовал школьников. Если система была слишкой сложной для детей, то она не допускалась к использованию в армии!

Далее последовательно будет рассмотрено много примеров криптосистем. Начнем с очень старой и имеющей ряд недостатков криптосистемы Цезаря. В разные времена использовалось много вариантов этой системы — они также будут обсуждены в следующем параграфе.

Не важно, как задается пространство исходных текстов. Метод Цезаря основан на *подстановках*: каждая буква заменяется другой буквой. Замена получается с помощью смещения от исходной буквы на k букв в алфавите. С конца алфавита следует циклический переход на его начало.

Таким образом, для $k = 3$ имеем следующие подстановки:

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
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

В этом случае исходный текст TRY AGAIN шифруется как WUB DJDLQ.

Ключевое пространство системы Цезаря состоит из 26 чисел: $0, 1, 2, \ldots, 25$. Алгоритм зашифрования E_k определяется ключом k как смещение вправо на k букв в алфавите. Соответственно алгоритм расшифрования D_k определяется как смещение влево на k букв в алфавите. Несколько иллюстраций:

$$E_{25}(\text{IBM}) = \text{HAL}, \quad E_6(\text{MUPID}) = \text{SAVOJ},$$

$$E_3(\text{HELP}) = \text{KHOS}, \quad E_1(\text{HOME}) = \text{IPNF},$$

$$D_6(\text{SAVOJ}) = E_{20}(\text{SAVOJ}) = \text{MUPID} \ .$$

Здесь можно установить несколько свойств таких алгоритмов зашифрования E и расшифрования D. Одним из них является *коммутативность*: если некоторые E и D применяются друг после друга, то порядок их использования не важен. К примеру,

$$E_3\, D_7\, E_6\, D_{11} = E_3\, E_6\, D_7\, D_{11} = D_9 = E_{17} \ .$$

Коммутативность будет являться основным свойством некоторых наших примеров. Для любого k, где $1 \le k \le 25$, также выполняются следующие соотношения:

$$D_k = E_{26-k}, \qquad D_k\, E_k = E_0 = D_0 \ .$$

Последнее равенство отражает тот факт, что E_k и D_k аннулируют друг друга.

Ключ расшифрования D_k может быть вычислен немедленно из ключа зашифрования E_k. Для любой криптосистемы D_k определяется (в математическом смысле) через E_k. Однако вычисление D_k из E_k может быть крайне трудным.

Для любой классической криптосистемы при оглашении алгоритма зашифрования E_k фактически раскрывается и алгоритм расшифрования D_k. Всякий, кто знает E_k, в состоянии вычислить и D_k. Конечно, вычисление может быть не таким быстрым, как для системы Цезаря, но очень часто оно может быть проведено за приемлемое время. Следовательно, E_k не может быть открыт.

Характеристическим свойством *криптосистем с открытым ключом* является то, что E_k может стать доступным без нарушения секретности. Ключи так искусно конструируются, что нахождение D_k из E_k, а также поиск исходного сообщения pt по заданным E_k и $E_k(pt)$, является трудновычислимым. В последующих главах это требование будет рассмотрено более подробно. Здесь мы только хотим обратить внимание на эту существенную особенность криптосистем с открытым ключом.

После обсуждения основ криптосистем переместимся в другой мир. Отныне мы будем называть перехватчика сообщений *криптоаналитиком*. Различие между криптоанализом и расшифрованием в том, что криптоаналитик справляется с задачей *без ключа расшифрования* D_k. В обоих случаях цель одинакова: восстановить исходное сообщение pt.

Представим рис. 1.1 более подробно (см. рис. 1.2).

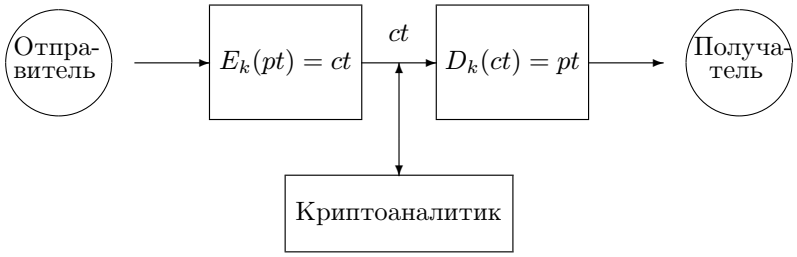


Рис. 1.2.

Отправитель (соответственно получатель) знает заранее E_k (соответственно D_k). Например, две стороны могут условиться о всех вопросах до отправки сообщения. Детали данного соглашения зависят от используемой криптосистемы. Эта процедура существенно различается для классических криптосистем и систем с открытым ключом.

Заметим, что для любого ключа k и исходного сообщения pt:

$$D_k(E_k(pt)) = D_k(ct) = pt \ .$$

Теперь сделаем несколько важных замечаний о криптоанализе. Начнем с формулирования следующего принципа:

Золотое правило для создателей криптосистем: Никогда не допускайте недооценки криптоаналитика.

Золотое правило следует применять ко всем действиям криптоаналитика: шпионство, изобретение методов атаки, эффективное использование вычислительной техники и т. д. Что касается информации об используемой системе, далее мы принимаем следующее соглашение: *криптоаналитик знает используемую криптосистему*. Это разумно, поскольку даже если криптоаналитику придется перебрать несколько криптосистем, сложность процедуры будет существенно той же, как и в случае, когда он работает с одной системой.

Хотя криптоаналитик знает криптосистему, он не знает ключа. Тем не менее если число всевозможных ключей мало, как в системе Цезаря, то все они могут быть проверены. (Отметим, что криптоаналитик имеет превосходные вычислительные средства!) Это означает, что криптосистема с малым числом ключей на практике оказывается бесполезной. Однако такие системы иногда могут применяться для иллюстрации специфических моментов, как в случае данного изложения.

Существенным условием для того, чтобы криптосистема была хорошей, является тот факт, что восстановление исходного сообщения pt из криптотекста ct трудновычислимо без знания алгоритма расшифрования D_k. Теперь обсудим более подробно возможные начальные условия криптоанализа. Укажем далее четыре основных условия. Также возможны некоторые симметричные модификации и комбинации основных условий. Они не будут обсуждаться ниже. Напомним, однако, что криптоаналитик всегда знает, какая криптосистема используется.

Условие (1): *Известен только криптотекст*. Здесь криптоаналитик должен исходить лишь из одного образца криптотекста. Для криптоаналитика всегда лучше иметь более длинный образец. В простых системах, таких, как система Цезаря, достаточно даже коротких образцов, так как обычно только один ключ будет использоваться для зашифрования осмысленного текста. В более сложных системах необходимы

длинные образцы криптотекстов. Эффективные криптоаналитические методы могут основываться на статистической информации о частоте появления букв в английском языке. Примеры будут даны позже.

Условие (2): *Известно некоторое исходное сообщение*. Здесь криптоаналитик знает заранее некоторую пару $(pt, E_k(pt))$, что может существенно помочь анализу заданного криптотекста ct. Очень простым примером опять является система Цезаря: любая пара любой длины дает ключ.

Условие (3): *Известно избранное исходное сообщение*. Криптоаналитик знает заранее некоторую пару $(pt, E_k(pt))$. Однако теперь pt выбран криптоаналитиком. В ситуациях, где криптоаналитик имеет определенные предположения о ключе, ясно, что это условие существенно лучше, чем (2). С другой стороны, данное условие, вероятно, будет более реалистичным, по крайней мере в тех случаях, когда криптоаналитик имеет возможность маскироваться под легального пользователя информационной системы.

Перед обсуждением условия (4) мы дадим пример криптосистемы, где начальное условие (3) часто дает наиболее лучшие возможности для криптоаналитика, чем начальное условие (2).

Пример 1.2. Криптосистема, разработанная Хиллом, базируется на линейной алгебре и интересна в историческом плане.

Пространства исходных сообщений и криптотекстов совпадают и равны Σ^*, где Σ — английский алфавит. Перенумеруем буквы в порядке их следования в алфавите: A получает номер 0, B — номер 1 и Z — номер 25.

Все арифметические операции выполняются по модулю 26 (числа букв в алфавите). Это означает, что 26 отождествляется с 0, 27 с 1, 28 с 2 и т. д.

Выберем целое число $d \geq 2$. Оно указывает размерность используемых матриц. В процедуре зашифрования наборы из d букв исходного сообщения шифруются вместе. Возьмем $d = 2$.

Пусть теперь M — квадратная $d \times d$–матрица. Элементами M являются целые числа от 0 до 25. Потребуем далее, чтобы матрица M была невырожденной, т.е. существовала бы M^{-1}. Например,

$$M = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \quad \text{и} \quad M^{-1} = \begin{pmatrix} 15 & 17 \\ 20 & 9 \end{pmatrix} .$$

Напомним, что арифметика ведется по модулю 26. Это дает, например,

$$2 \cdot 17 + 5 \cdot 9 = 79 = 1 + 3 \cdot 26 = 1 ,$$

как и должно быть, единицу на главной диагонали единичной матрицы.

Зашифрование осуществляется с помощью уравнения

$$MP = C \ ,$$

где P и C — d–размерные вектор-столбцы. Более подробно: каждый набор из d букв исходного сообщения определяет вектор P, компонентами которого являются номера букв. Наконец, C опять интерпретируется как набор d букв криптотекста.

Например, исходное сообщение HELP определяет два вектора

$$P_1 = \begin{pmatrix} H \\ E \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix} \quad \text{и} \quad P_2 = \begin{pmatrix} L \\ P \end{pmatrix} = \begin{pmatrix} 11 \\ 15 \end{pmatrix} \ .$$

Из уравнений

$$MP_1 = \begin{pmatrix} 7 \\ 8 \end{pmatrix} = C_1 \quad \text{и} \quad MP_2 = \begin{pmatrix} 0 \\ 19 \end{pmatrix} = C_2$$

получаем криптотекст HIAT.

Рассмотрим теперь сферу деятельности нашего криптоаналитика. Предположим, что аналитик догадался, что $d = 2$. Ему нужно найти матрицу M или, еще лучше, обратную матрицу M^{-1}. Для этой цели он выбирает исходное сообщение HELP и узнает, что соответствующий криптотекст есть HIAT. Криптоаналитик знает, что

$$M\begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \end{pmatrix} \quad \text{и} \quad M\begin{pmatrix} 11 \\ 15 \end{pmatrix} = \begin{pmatrix} 0 \\ 19 \end{pmatrix} \ .$$

Это может быть записано в виде

$$M = \begin{pmatrix} 7 & 0 \\ 8 & 19 \end{pmatrix} \begin{pmatrix} 7 & 11 \\ 4 & 15 \end{pmatrix}^{-1} = \begin{pmatrix} 7 & 0 \\ 8 & 19 \end{pmatrix} \begin{pmatrix} 19 & 19 \\ 14 & 21 \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \ .$$

Обратная матрица M^{-1} сразу же вычисляется из M. После этого любой криптотекст может быть расшифрован с помощью M^{-1}.

Важным моментом в этих вычислениях является существование обратной матрицы к $\begin{pmatrix} 7 & 11 \\ 4 & 15 \end{pmatrix}$. С другой стороны, наш криптоаналитик выбрал исходное сообщение HELP, порождающее матрицу $\begin{pmatrix} 7 & 11 \\ 4 & 15 \end{pmatrix}$. Значит, он осуществляет выбор таким образом, чтобы результирующая матрица имела обратную.

Предположим теперь, что криптоаналитик работает с другой начальной постановкой — “известно некоторое исходное сообщение”. Более подробно: пусть криптоаналитик знает, что CKVOZI — криптотекст, соответствующий исходному сообщению SAHARA. Хотя мы имеем здесь более длинный пример сообщения, чем ранее, однако извлекаемая из него информация намного скуднее.

Действительно, теперь уравнения для сообщения-криптотекста имеют вид

$$M\begin{pmatrix}18\\0\end{pmatrix}=\begin{pmatrix}2\\10\end{pmatrix},\quad M\begin{pmatrix}7\\0\end{pmatrix}=\begin{pmatrix}21\\14\end{pmatrix}\quad\text{и}\quad M\begin{pmatrix}17\\0\end{pmatrix}=\begin{pmatrix}25\\8\end{pmatrix}.$$

Не существует обратимой квадратной матрицы, которая может быть образована из трех векторов-столбцов, появляющихся как коэффициенты M.

Криптоаналитик обнаруживает, что любая обратимая квадратная матрица

$$M'=\begin{pmatrix}3&x\\2&y\end{pmatrix}$$

может быть базисом криптосистемы, потому что она шифрует SAHARA как CKVOZI. Таким образом, криптоаналитик может остановиться на матрице

$$M'=\begin{pmatrix}3&1\\2&1\end{pmatrix},$$

для которой обратной является матрица

$$(M')^{-1}=\begin{pmatrix}1&25\\24&3\end{pmatrix}.$$

Криптоаналитик готов к перехвату криптотекста. Он получает текст NAFG и тут же вычисляет

$$\begin{pmatrix}1&25\\24&3\end{pmatrix}\begin{pmatrix}13\\0\end{pmatrix}=\begin{pmatrix}13\\0\end{pmatrix}\quad\text{и}\quad\begin{pmatrix}1&25\\24&3\end{pmatrix}\begin{pmatrix}5\\6\end{pmatrix}=\begin{pmatrix}25\\8\end{pmatrix}.$$

Два вектора-столбца порождают исходное сообщение NAZI. Однако легальный пользователь знает оригинальную матрицу M и ее обратную матрицу и вычисляет

$$\begin{pmatrix}15&17\\20&9\end{pmatrix}\begin{pmatrix}13\\0\end{pmatrix}=\begin{pmatrix}13\\0\end{pmatrix}\quad\text{и}\quad\begin{pmatrix}15&17\\20&9\end{pmatrix}\begin{pmatrix}5\\6\end{pmatrix}=\begin{pmatrix}21\\24\end{pmatrix},$$

дающие исходное сообщение NAVY.

Наш криптоаналитик сделал грубую ошибку, которая могла привести к ложным действиям!

□

Продолжим наш список возможных начальных условий криптоанализа.

Условие (4): *Известен ключ зашифрования*. Криптоаналитик знает алгоритм зашифрования E_k и старается найти соответствующий алгоритм расшифрования D_k до реального получения любого образца криптотекста.

Условие (4) очень типично для криптосистем с открытым ключом. Алгоритм зашифрования E_k может быть опубликован заранее, и может пройти несколько месяцев до того, как E_k будет использован для шифрования важных сообщений. Таким образом, криптоаналитик обычно имеет много времени для *предварительного криптоанализа* (поиска алгоритма расшифрования), в то время как при появлении сообщений ему приходится действовать в условиях недостатка времени. Особенно ценным является завершение поиска в период, когда "время дешево".

В некоторых криптосистемах с открытым ключом невозможно построить D_k из E_k, потому что нельзя распознать верное D_k среди нескольких вариантов. Для этой цели необходимо иметь несколько образцов криптотекста. В других криптосистемах с открытым ключом алгоритм зашифрования D_k может быть найден из E_k, если только очень сильно повезет, например, угадать множители из произведения двух больших простых чисел.

1.2. Одноалфавитные системы

В этой главе обсуждаются *классические* криптосистемы, в отличие от криптосистем *с открытым ключом*, и закладываются основы, необходимые для чтения основных частей книги. Здесь мы уделим внимание обоим мирам криптографии.

Напомним о различии между классическими криптосистемами и криптосистемами с открытым ключом. В первых ключ расшифрования D_k легко может быть вычислен из ключа зашифрования E_k, в то время как в криптосистеме с открытым ключом E_k может быть опубликован без потери секретности D_k. По этой причине классические системы часто также называют как *симметричные*, или *двусторонние*, а системы с открытым ключом — как *несимметричные*, или *односторонние*.

Обсудим несколько важных положений. До сих пор мы не комментировали условие (3) для хорошей криптосистемы, выдвинутое сэром

Фрэнсисом Беконом: криптотекст не должен вызывать подозрений, т.е. должен выглядеть естественно.

В наши дни и криптотекст, и исходное сообщение — это последовательность непонятных на первый взгляд битов, поэтому данное требование уже не является важным. Последовательность битов обычно не более невинна, чем другая последовательность! Тем не менее в прошлом это требование очень часто принималось в расчет.

Лучшим методом, удовлетворяющим данному требованию, является алгоритм *"посреди мусора"*. Реальное сообщение (шифрованное или нет) пополняется "мусорными буквами", которые совершенно не относятся к сообщению, но делается это так, чтобы все сообщение выглядело невинно.

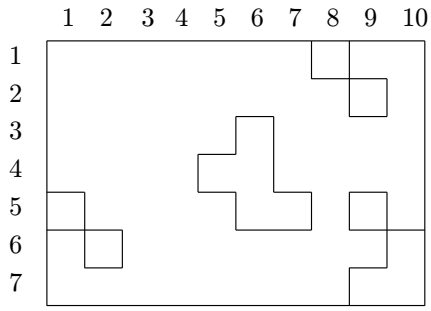


Рис. 1.3.

Ришелье использовал листы картона с прорезями. Значимыми являлись только буквы, видимые через прорези. И отправитель, и получатель имели одинаковые листки. Один такой лист изображен на рис. 1.3. Лист покрывает участок текста, заключенного в прямоугольник с семью строками и десятью столбцами, всего 70 символов текста. Для длинных сообщений листок применялся несколько раз. Прорези размещаются в позициях

$$(1,8),\ (2,9),\ (3,6),\ (4,5),\ (4,6),\ (5,1),\ (5,6)\ ,$$

$$(5,7),\ (5,9),\ (6,2),\ (6,10),\ (7,9),\ (7,10)\ .$$

Следующий текст выглядит как невинное любовное письмо:

I		L	O	V	E		Y	O	U
I		H	A	V	E		Y	O	U
D	E	E	P		U	N	D	E	R
M	Y		S	K	I	N		M	Y
L	O	V	E		L	A	S	T	S
F	O	R	E	V	E	R		I	N
H	Y	P	E	R	S	P	A	C	E

Однако, используя криптосистему Ришелье (листок на рис. 1.3), получим зловещую команду: YOU KILL AT ONCE.

Существует много *классификаций криптосистем*, некоторые из которых будут упомянуты. Принципы классификации относятся не к качеству криптосистем (хорошая или плохая), а говорят о присущих свойствах при их проектировании.

Очень старая классификация — системы *подстановок* и *перестановок*, часто называемых также перемещениями. К примеру, [Ga] говорит о *замещающих* и *перестановочных шифрах*.

В первых из них буквы исходного сообщения заменяются на подстановки. Замены расположены в криптотексте в том же порядке, что и в оригинале. Если использование замен сохраняется постоянным на протяжении всего текста, то криптосистема называется *одноалфавитной*. Этот термин выражает идею, что здесь имеется только одна последовательность замещающих букв: всякая буква исходного сообщения одинаково заменяется на протяжении всего текста. Если исходное сообщение написано на любом естественном языке, криптоанализ всегда может основываться на статистическом распределении букв. Примеры будут рассмотрены ниже.

Одноалфавитные системы замен контрастируют с *многоалфавитными* криптосистемами, где использование подстановок меняется в различных частях текста. Мы вернемся к многоалфавитным криптосистемам в параграфе 1.3. Наиболее распространенные криптоаналитические методы связаны с многоалфавитными системами.

В криптосистемах с перестановками (или транспозициями) буквы исходного сообщения переупорядочиваются. Порядок перестановки при этом может комбинироваться и с некоторой другой идеей.

Рассмотрим пример системы с перестановкой. Исходное сообщение поделено на блоки по три буквы в каждом. Перестановка букв в каждом блоке осуществляется таким образом, что первая буква становится третьей, а вторая и третья буквы сдвигаются на один шаг назад. К примеру, исходное сообщение LETUSGOTOFRANCE станет ETLSGUTOORAFCEN (напомним,что пробелы между отдельными словами часто игнорируются).

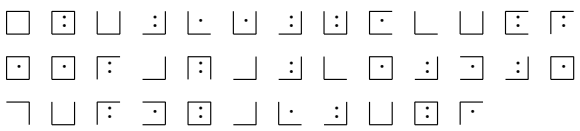
В этом параграфе обсуждаются одноалфавитные системы. В качестве пространства исходных сообщений выбрано множество слов, составленных из букв английского алфавита. Поэтому, каждая буква A, B, C,..., Z заменяется на $x_1, x_2, x_3, \ldots, x_{26}$ везде в исходном сообщении. Подстановки для них различны, но они могут содержать буквы, не входящие в английский алфавит. Крайним является случай, когда криптотекст содержит некоторые совершенно отличные от букв символы. В качестве примера рассмотрим следующее соответствие для букв английского алфавита:

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	

Линии, окаймляющие каждую букву вместе с точками (две, одна или ни одной), указывают замену для буквы. Так, исходное сообщение WE TALK ABOUT FINNISH SAUNA MANY TIMES LATER будет зашифровано как



На первый взгляд кажется, что мы можем сказать довольно мало об одноалфавитных системах. Если исходное сообщение написано на английском или другом естественном языке, то статистический анализ вскроет эту систему. Если образец текста достаточно длинен, то наиболее часто встречающийся в криптотексте символ представляет собой наиболее употребительную букву естественного языка. Достаточно обычно отыскать таким методом несколько букв, а об остальных затем можно догадаться. С другой стороны, если пространство исходных сообщений есть Σ^*, где Σ — английский алфавит, и нет никакой дополнительной информации, то криптоанализ одноалфавитной системы невозможен. Нельзя найти соответствие между буквами исходного сообщения и их заменами: все соответствия равновероятны. На самом деле, в этом случае одноалфавитная шифровка является просто кодом; правильная шифровка имела бы место в том случае, если значимые сообщения были бы переведены (с одинаковым распределением) в слова из Σ^*.

При таком первоначальном рассмотрении одноалфавитных систем упускается несколько важных моментов. В действительности об одноалфавитных системах может быть сказано много. Основной вопрос

заключается в *управлении ключом*: все раскрывается, если станет известно соответствие между исходными буквами и их заменой (т.е. известен ключ). Следовательно, ключ не должен быть доступен ни в какой форме: ни в письменном виде, ни в памяти компьютера. Отправитель и получатель хранят ключ в памяти. Различные способы хранения приводят к разным одноалфавитным системам. Рассмотрим некоторые из них.

Мы уже говорили о системе Цезаря в параграфе 1.1. Замена буквы осуществляется сдвигом на k шагов в алфавите. В системе Цезаря и других подобных системах будет использоваться *числовое кодирование* букв:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Так, согласно системе Цезаря, каждая буква α становится $\alpha + k$. Вся арифметика здесь ведется по модулю 26. Ни кодирование, ни декодирование (из чисел в буквы) не относится к самому процессу зашифрования.

Число всех возможных ключей в системе Цезаря очень мало. Другим крупным недостатком с точки зрения безопасности является сохранение алфавитного порядка и в последовательности подставляемых букв; изменяются только начальные позиции. В рассмотренных ниже аффинных криптосистемах этот недочет устранен.

Отступление: Старые времена. Юлий Цезарь повествует о посылке зашифрованного сообщения Цицерону. Используемая при этом система подстановок была одноалфавитной, но не являлась системой Цезаря: латинские буквы заменялись на греческие способом, который не был ясен из рассказа Цезаря. Информация о том, что Цезарь действительно использовал криптосистему Цезаря, пришла от Светония. В действительности, согласно Светонию, сдвиг в алфавите осуществлялся на три буквы. Информация о том, что Цезарь использовал и другие сдвиги, документально не подтверждена.

Система Цезаря не является старейшей. Возможно, что наиболее древней из известных является система греческого историка Полибия, умершего за тридцать лет до рождения Цезаря. Не известно, использовал ли Полибий свою систему для криптографических целей. Опишем эту систему для английского алфавита, в котором опущена буква J.

Рассмотрим следующий квадрат, часто называемый в наши дни доской Полибия:

	A	B	C	D	E
A	A	B	C	D	E
B	F	G	H	I	K
C	L	M	N	O	P
D	Q	R	S	T	U
E	V	W	X	Y	Z

Каждая буква α может быть представлена парой букв, указывающих на строку и столбец, в которых расположена данная буква α. Так, представления для букв K, O и T есть BE, CD и DD соответственно. Исходное сообщение LETUSGOTOSAUNA шифруется как

CAAEDDDEDCBBCDDDCDDCAADECCAA.

В нашей терминологии система Полибия — это одноалфавитная система замен в алфавите из 25 букв { AA, AB, . . . , AE, BA,. . . , EE }.

Совместно с криптографией часто использовалось искусство *тайнописи* (сокрытия сообщений). К примеру, зашифрованное сообщение может быть написано бесцветными чернилами. Древнегреческий историк Геродот не упоминает о криптосистемах в нашем смысле этого слова, зато у него имеются рассказы о тайнописи. Вот один из них.

Гистай и его зять Аристагор предварительно договорились, что содержащее несколько точек сообщение будет означать, что Аристагор должен поднять мятеж против Персии. Когда Гистай решил действительно послать такое сообщение Аристагору, он обнаружил, что близлежащая территория тщательно охраняется. Тогда Гистай побрил голову своему наиболее верному рабу, нарисовал точки на голове и подождал, пока волосы отрастут вновь. Когда это свершилось, он послал раба со следующей запиской для Аристагора: “Побрей мою голову!”

Вышеизложенная история говорит нам и о том, что в те времена криптографы не были так ограничены во времени, как сегодня.

□

Аффинная криптосистема определяется двумя целыми числами a и b, где $0 \le a, b \le 25$, a и 26 взаимно просты. Заменой для буквы α будет $a\alpha + b \pmod{26}$. Здесь мы работаем с числовыми кодами букв и все арифметические действия выполняются по модулю числа 26. К примеру, если $a = 3$ и $b = 5$, то получаем следующее соответствие для числовых кодов букв:

0	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
5	8	11	14	17	20	23	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2

Когда декодируем числа в буквы, получим следующее соответствие для букв:

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
F	I	L	O	R	U	X	A	D	G	J	M	P	S	V	Y	B	E	H	K	N	Q	T	W	Z	C

Исходное сообщение NOTEVERYSTEAMBATHISSAUNA зашифруется в SVKRQREZHKRFPIFKADHHFNSF.

Условие взаимной простоты пары чисел a и 26 необходимо для обеспечения биективности отображения $f(\alpha) = a\alpha + b$. Если мы рассмотрим отображение $10\alpha + 1$, где данное условие не выполняется, то буквы A и N обе отображаются в B и, следовательно, B может быть расшифрована и как A, и как N. С другой стороны, нет числового кода отображаемого в O, и, следовательно, O не требуется в алфавите подстановок. Легко найти все пары букв, отображаемых в одну и ту же букву так же, как и все буквы, не требующиеся в алфавите подстановок.

Теперь вновь перейдем в мир криптоаналитика.

Пример 1.3. Исходное сообщение, составленное из букв английского алфавита, разбивается на блоки по пять букв в каждом, а затем шифруется с помощью аффинной системы. Пробелы между словами в исходном сообщении при этом игнорируются. Пусть получен следующий криптотекст:

B	H	J	U	H		N	B	U	L	S		V	U	L	R	U		S	L	Y	X	H
O	N	U	U	N		B	W	N	U	A		X	U	S	N	L		U	Y	J	S	S
W	X	R	L	K		G	N	B	O	N		U	U	N	B	W		S	W	X	K	X
H	K	X	D	H		U	Z	D	L	K		X	B	H	J	U		H	B	N	U	O
N	U	M	H	U		G	S	W	H	U		X	M	B	X	R		W	X	K	X	L
U	X	B	H	J		U	H	C	X	K		X	A	X	K	Z		S	W	K	X	X
L	K	O	L	J		K	C	X	L	C		M	X	O	N	U		U	B	V	U	L
R	R	W	H	S		H	B	H	J	U		H	N	B	X	M		B	X	R	W	X
K	X	N	O	Z		L	J	B	X	X		H	B	N	F	U		B	H	J	U	H
L	U	S	W	X		G	L	L	K	Z		L	J	P	H	U		U	L	S	Y	X
B	J	K	X	S		W	H	S	S	W		X	K	X	N	B		H	B	H	J	U
H	Y	X	W	N		U	G	S	W	X		G	L	L	K							

Прежде чем применять какие-либо специальные криптоаналитические методы, сделаем несколько общих замечаний. Все наши примеры недостаточны с точки зрения реальной криптографии. Образцы текста

очень коротки, а используемые числа малы. Причина этого заключается в том, что если мы попытаемся описать реальную ситуацию, то изложение станет трудным для восприятия.

Сколько в аффинной системе имеется различных ключей? Каждый ключ полностью определен парой целых чисел a и b, задающих отображение $a\alpha + b$. Существует 12 значений для a: 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25. Существует 26 возможных значений для b, причем они могут быть использованы независимо от значений для a, за исключением случая $a = 1, b = 0$. Это дает в совокупности $12 \cdot 26 - 1 = 311$ возможных ключей.

Перебрать и проверить все 311 ключей очень легко с помощью компьютера, и, следовательно, проведение криптоанализа не составит труда. Тем не менее мы хотим упростить этот изнурительный поиск, что приобретает исключительную значимость в более запутанных криптоаналитических задачах.

Основная криптоаналитическая атака против системы замен начинается с *подсчета частот* появления символов: определяется число появлений каждой буквы в криптотексте. Распределение букв в криптотексте сравнивается затем с распределением букв в алфавите исходных сообщений, к примеру, в английском. Буква с наивысшей частотой в криптотексте будет заменяться на букву с наивысшей частотой в английском и т.д. Вероятность успешного вскрытия системы повышается с увеличением длины криптотекста.

Информация о распределении букв английского алфавита содержится в множестве различных таблиц, аналогичные таблицы имеются и для других естественных языков. Подчеркнем тем не менее, что ни одна из этих таблиц не содержит окончательной информации. Даже порядок букв относительно их частоты появления в тексте отличен для разных таблиц. Распределение букв очень сильно зависит от типа текста: обыкновенная проза, разговорный язык, технический, язык телекоммуникаций и т.д. Нет таблицы, которая может учесть предположительно все типы текстов!

Однако есть некоторые вещи, общие для всех таблиц, описывающих английский язык. Буква E всегда возглавляет список частот, а T идет второй. Почти всегда A или O на третьей позиции. Кроме того, девять букв E, T, A, O, N, I, S, R, H всегда имеют частоту выше, чем любые другие буквы. Эти девять букв заполняют 70% английского текста. Читателю предоставляется возможность написать разумный длинный отрывок текста на английском, где буквы с высокой частотой не составляют большинства!

Что касается позиционной частоты, буквы A, I, H не часто стоят в

конце слова, в то время как E, N, R появляются с гораздо меньшей частотой в начальной позиции, чем в конечной. Оставшиеся буквы T, O, S из высокочастотного класса появляются с одинаковой вероятностью и в начале, и в конце. Такие наблюдения, касающиеся позиционной частоты, конечно, не имеют силы для рассматриваемого примера, так как деление на блоки исходного сообщения уничтожает начальные и конечные позиции.

В следующей таблице буквы английского алфавита упорядочены по частоте их появления. Также указаны и проценты для каждой буквы. Таблица взята из [Ga].

Высокий:	
	%
E	12.31
T	9.59
A	8.05
O	7.94
N	7.19
I	7.18
S	6.59
R	6.03
H	5.14
	70.02

Средний:	
	%
L	4.03
D	3.65
C	3.20
U	3.10
P	2.29
F	2.28
M	2.25
W	2.03
Y	1.88
	24.71

Низкий:	
	%
B	1.62
G	1.61
V	.93
K	.52
Q	.20
X	.20
J	.10
Z	.09
	5.27

В нашем примере исходное сообщение написано на английском. Однако, ради сравнения, в следующей таблице представлены наиболее часто встречающиеся буквы в других языках.

Английский	%
E	12.31
T	9.59
A	8.05
O	7.94
N	7.19
I	7.18
S	6.59
R	6.03
H	5.14

Немецкий	%
E	18.46
N	11.42
I	8.02
R	7.14
S	7.04
A	5.38
T	5.22
U	5.01
D	4.94

Финский	%
A	12.06
I	10.59
T	9.76
N	8.64
E	8.11
S	7.83
L	5.86
O	5.54
K	5.20

Французский	%
E	15.87
A	9.42
I	8.41
S	7.90
T	7.26
N	7.15
R	6.46
U	6.24
L	5.34

Итальянский	%
E	11.79
A	11.74
I	11.28
O	9.83
N	6.88
L	6.51
R	6.37
T	5.62
S	4.98

Испанский	%
E	13.15
A	12.69
O	9.49
S	7.60
N	6.95
R	6.25
I	6.25
L	5.94
D	5.58

Заметим, что буквы I, N, S, E, A появляются в высокочастотном классе каждого языка!

Все эти общие замечания подходят к нашему первому длинному криптоаналитическому примеру. Возвратимся теперь к криптотексту, считая число появлений каждой буквы:

Высокий:	
	Число
X	32
U	30
H	23
B	19
L	19
N	16
K	15
S	15
W	14
	183=78.21%

Средний:	
	Число
J	11
O	6
R	6
G	5
M	4
Y	4
Z	4
C	3
A	2
	45=19.23%

Низкий:	
	Число
D	2
V	2
F	1
P	1
E	0
I	0
Q	0
T	0
	6=2.56%

Частота букв X, U, H, B, L, N, K, S, W даже выше,чем частота букв E, A, T, O, N, I, S, R, H. Поэтому первые буквы соответствуют последним. Так как мы связаны с аффинной системой, достаточно найти *корректную замену для двух букв*.

Сделаем попытку для первых двух самых высокочастотных букв: X — замена для E, U — для T. Аффинная система отображает каждый числовой код α в $a \cdot \alpha + b$. Следовательно,

$$4a + b \equiv 23 \pmod{26} \quad \text{и} \quad 19a + b \equiv 20 \pmod{26} .$$

Эти сравнения по модулю 26 имеют единственное решение:

$$a = 5 \qquad \text{и} \qquad b = 3 .$$

Для отображения $5\alpha + 3$ построим следующую таблицу перевода криптотекста в исходное сообщение.

Криптотекст	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
Сообщение	P	K	F	A	V	Q	L	G	B	W	R	M	H	C	X	S	N	I	D	Y	T	O	J	E	Z	U

Применяя эту таблицу к нашему криптотексту, мы получаем следующее исходное сообщение:

K G W T G C K T M D ...

Эта бессмыслица выглядит не очень обещающей. В английском есть некоторые гласные! Давайте попытаемся еще раз.

Теперь предположим, что самая высокочастотная буква E отображается в X. А вместо второй наиболее встречающейся буквы рассмотрим третью: пусть A отображается в H. Это даст нам сравнения

$$4a + b \equiv 23 \pmod{26} \quad \text{и} \quad b \equiv 7 \pmod{26} .$$

Существует два решения для a: $a = 4$ и $a = 17$. Однако первое не подходит (так как a и 26 взаимно просты), и искомым отображением будет $17 \cdot \alpha + 7$. Получаем следующую таблицу перевода:

Криптотекст	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
Сообщение	V	S	P	M	J	G	D	A	X	U	R	O	L	I	F	C	Z	W	T	Q	N	K	H	E	B	Y

Это даст исходное сообщение:

S A U N A I S N O T K N O W N T O B E A
F I N N I S H I N V E N T I O N B U T T
H E W O R D I S F I N N I S H T H E R E
A R E M A N Y M O R E S A U N A S I N F
I N L A N D T H A N E L S E W H E R E O
N E S A U N A P E R E V E R Y T H R E E
O R F O U R P E O P L E F I N N S K N O
W W H A T A S A U N A I S E L S E W H E
R E I F Y O U S E E A S I G N S A U N A
O N T H E D O O R Y O U C A N N O T B E
S U R E T H A T T H E R E I S A S A U N
A B E H I N D T H E D O O R

Намного лучше! Нам осталось расставить пробелы и знаки пунктуации: Sauna is not known to be a Finnish invention but the word is Finnish. There are many more saunas in Finland than elsewhere: one sauna per every

three or four people. Finns know what a sauna is. Elsewhere if you see a sign “sauna” on the door, you cannot be sure that there is sauna behind the door[2].

Читатель может проверить точное соответствие последовательностей букв из высокочастотного класса, однако буквы C и M исходного сообщения из среднего класса меняются на буквы B и V из низкочастотного класса. Это неудивительно, так как в исходном сообщении длины 234 среднеожидаемые частоты этих букв лежат в диапазоне от 2 до 7, где небольшие изменения в ожидаемых значениях могут произойти только “локально” с помощью одного или двух специфических слов.

Добавим несколько слов о содержимом исходного сообщения. Криптоаналитик не знал, что много слов в нашем примере связано с сауной. Иначе он мог просто подставить SAUNA для повторяющейся буквенной комбинации BHJUH!

□

На этом завершим обсуждение аффинных систем с точки зрения криптосистемы и криптоанализа. Аффинные системы, которые использовались на практике несколько веков назад, сегодня применяются только для иллюстрации основных криптографических положений. Естественным математическим обобщением подобных систем являются *полиномиальные криптосистемы*: вместо линейной функции $f(x) = a \cdot \alpha + b$ выбирается произвольный многочлен. Однако полиномиальные системы малоинтересны с точки зрения криптографии. Напомним, что главной мотивацией для аффинных систем является управление ключом: мы хотим представить ключи зашифрования и расшифрования в компактной форме. Ключ всегда состоит из последовательности 26 букв. Представление в виде полинома может быть таким сложным, как и тривиальное представление в виде исходной последовательности.

Обсудим следующую одноалфавитную систему, называемую системой Цезаря с ключевым словом (KEYWORD-CAESAR). Выберем число k, $0 \leq k \leq 25$, и слово или короткое предложение в качестве *ключевого слова*. Все буквы в ключевом слове должны быть различными. Пусть в качестве такого слова выбрано HOW MANY ELKS и число 8.

Ключевое слово напишем теперь внизу под буквами алфавита, начиная с буквы, числовой код которой совпадает с выбранным числом k:

[2]Сауна не является финским открытием, но слово финское. В Финляндии существует больше саун, чем где-нибудь еще: одна сауна на 3 или 4 человека. Финны знают, что такое сауна. Если вы увидите слово “сауна” на двери где-нибудь в другом месте, вы не можете быть уверены, что за дверью находится именно сауна.

0 8 25
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
H O W M A N Y E L K S

Оставшиеся буквы записываются в алфавитном порядке после ключевого слова:

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
P Q R T U V X Z H O W M A N Y E L K S B C D F G I J

Теперь мы имеем подстановку для каждой буквы. Исходное сообщение ERROLFLYNN шифруется как UKKYMVMINN.

Требование, чтобы все буквы ключевого слова были различными, не обязательно. Мы можем просто записывать ключевое слово без повторения одинаковых букв. Для примера, ключевое слово ENGLAND EXPECTS EVERY MAN TO DO HIS DUTY и число 2 порождают следующую таблицу перевода:

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
W Z E N G L A D X P C T S V R Y M O H I U B E J K Q

Количество ключей в системе Цезаря с ключевым словом огромно. Хотя, может быть, и невозможно найти ключевые слова для всех 26! возможных комбинаций 26 букв, это может быть сделано для значительно больших подклассов. Теперь рассмотрим данную систему с точки зрения криптоаналитика.

Пример 1.4. Система Цезаря с ключевым словом (возможно, с повторениями в ключевом слове) использовалась для получения следующего криптотекста, где также сохранены оригинальные пробелы между словами исходного сообщения на английском:

T I V D Z C R T I C F Q N I Q T U T F
Q X A V F C Z F E Q X C P C Q U C Z W K
Q F U V B C F N R R T X T C I U A K W T Y
D T U P M C F E C X U U V U P C B V A N H C
V R U P C F E Q X C U P C F U V B C
X V I U Q T I F F U V I C F N E N Q A A K
V I U P C U V E U V U Q G C Q F Q N I Q
W Q U P T U I F Q A F V I C X C F F Q M K
U P Q U U P C F U V B C T F E M V E C M A K
P C Q U C Z Q I Z U P Q U K V N P Q B C
U P C R Q X T A T U K V R U P M V D T I Y
D Q U C M V I U P C F U V I C F

Подсчет частот дает следующее распределение среди 241 букв:

Высокие:	
	Число
U	32
C	31
Q	23
F	22
V	20
P	15
T	15
I	14
A	8
	180=74.69%

Средние:	
	Число
X	8
K	7
N	7
E	6
M	6
R	6
B	5
Z	5
D	4
	54=22.41%

Низкие:	
	Число
W	3
Y	2
G	1
H	1
J	0
L	0
O	0
S	0
	7=2.90%

Сравнивая частоту A с частотами из средней группы, мы видим, что любая буква из средней группы может быть среди высокочастотных букв E, T, A, O, N, I, S, R, H. Кроме того, частоты в конце последней группы не дают информации, так как текст является небольшим. Следовательно, мы можем начать поиск соответствия с высокочастотных букв, за исключением A. Пара попыток даст правильный выбор, после которого оставшиеся буквы с некоторым числом появлений могут быть установлены по смыслу.

Тем не менее существует и более короткий путь, который делает задачу анализа текста очень легкой. Этот путь показывает, как опасно сохранять оригинальные пробелы в криптотексте.

Криптотекст содержит однобуквенные слова T и Q. Они должны быть A и I. Так как слово T встречается один раз, а Q — три раза, то T — есть I, а Q — есть A. В этом можно быть почти уверенным, посмотрев на частоту появления букв T и Q.

Трехбуквенное слово UPC встречается 7 раз, в то время как другие трехбуквенные слова — по одному разу. UPC должно быть THE, что подтверждается частотным анализом.

Теперь мы можем расшифровать буквы C, P, Q, T, U из высокочастотной группы. Продолжение поиска будет легким. Из слов TU TF (встречается дважды!) узнаем, что F — есть S и из слова UV, что V — есть O. Слово VI и тот факт, что I имеет высокую частоту, указывает на то, что I есть N — при этом предположение, что I есть R, опровергается словом XVIUQTIF.

После расшифрования восьми из девяти высокочастотных букв мы имеем множество слов в криптотексте с одной неизвестной буквой. Это приводит к однозначному расшифрованию оставшихся букв. В итоге имеем следующую таблицу расшифрования:

Криптотекст	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
Сообщение	L	V	E	W	P	S	K	M	N	?	Y	?	R	U	?	H	A	F	?	I	T	O	B	C	G	D

Напишем исходное сообщение, используя знаки пунктуации.

I now define sauna. It is a closed space heated by a stove sufficiently big with respect to the volume of the space. The stove contains stones, usually on the top. To take a sauna bath it is also necessary that the stove is properly heated and that you have the facility of throwing water on the stones.

Преобразуем таблицу расшифрования в таблицу зашифрования с помощью упорядочивания букв исходного сообщения в алфавитном порядке:

Сообщение	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
Криптотекст	Q	W	X	Z	C	R	Y	P	T	?	G	A	H	I	V	E	?	M	F	U	N	B	D	?	K	?

Следовательно, ключевым словом является CRYPTOGRAPHY GIVES ME FUN, начинающееся с позиции 4. Буквы J, Q, X, Z, отсутствующие в исходном сообщении, должны зашифроваться как O, S, J, L соответственно. Отметим, наконец, что высокочастотная буква R английского алфавита отсутствует в классе высокочастотных букв нашего примера.

□

Простейшая защита против атак, основанных на подсчете частот, обеспечивается в криптосистеме омофонов (HOMOPHONES), которая также является одноалфавитной: только при этом буквы исходного сообщения имеют несколько замен. Их число пропорционально частоте появления буквы. Так, английская буква E имеет 3 замены для каждой подстановки буквы L, и 123 замены для каждой подстановки буквы J. Шифруя букву исходного сообщения, мы выбираем случайно одну из ее замен. (Мы следуем таблице распределения из примера 1.3.) Поэтому алгоритм зашифрования не является функцией.

Замены (часто называемые омофонами) могут быть представлены трехразрядными числами от 000 до 999. Мы присвоили E случайно 123 таких номеров. J и Z получают по одному номеру, а B и G — по 16 номеров. Если омофоны присваиваются случайно различным появлениям одной и той же буквы, каждый омофон появляется в криптотексте равновероятно.

Следовательно, простой подсчет частот ничего не даст криптоаналитику. Однако доступна также информация о распределении пар и троек букв в различных естественных языках. Криптоанализ, основанный на такой информации, будет более успешным.

1.3. Многоалфавитные и другие системы

Напомним, что криптосистема называется *одноалфавитной*, если замены одинаковы на протяжении всего текста. Одноалфавитным системам противопоставлены *многоалфавитные*: использование замен не одинаково для различных частей текста.

Используются ли замены для индивидуальных букв или, скажем, для пар букв? Этот вопрос возникает только в том случае, если система оперирует с основным алфавитом, элементами которого являются упорядоченные пары английских букв. Если замена для такой пары всегда одна и та же, мы называем систему одноалфавитной.

В параграфе 1.2 наши примеры одноалфавитных подстановок связаны с индивидуальными буквами и заменами для них. Таким образом, системы были одноалфавитными в очень строгом смысле. Теперь рассмотрим криптосистему, основанную на заменах для пар букв, где замена для каждой пары сохраняется на протяжении всего текста. Такая система может быть определена как одноалфавитная “в широком смысле”. Позже в этом параграфе мы обсудим многоалфавитные системы, которые не являются одноалфавитными даже в широком смысле.

Напомним систему Хилла, рассмотренную в примере 1.2. Если размерность матриц равна двум, то мы шифруем пары букв. Хотя буква A может быть зашифрована по-разному в различных парах исходного сообщения, пары, такие, как AL, будут шифроваться всегда одинаково, тогда как появление этой пары в CALL будет шифроваться иначе, так как AL не возникает при делении на блоки. В любом случае система Хилла является одноалфавитной в широком смысле. Простого подсчета частот будет недостаточно для криптоанализа. Здесь необходимы более продуманные методы подсчета частот, в частности такие, как статистический анализ пар букв. Эта задача будет обсуждаться в примере 1.5.

Система Плейфейра (PLAYFAIR), которую мы рассмотрим сейчас, названа в честь барона Плейфейра. Буквы английского алфавита, где отсутствует J, упорядочиваются в квадрате 5×5, например:

S	Y	D	W	Z
R	I	P	U	L
H	C	A	X	F
T	N	O	G	E
B	K	M	Q	V

Квадрат является основой для зашифрования (и расшифрования) согласно следующим правилам:

1. Исходный текст делится на блоки по две буквы в каждом. Текст имеет четную длину и в нем не должно быть блоков, содержащих две одинаковые буквы. Если эти требования не выполнены, то текст модифицируется. Возможно, придется даже сделать незначительные орфографические ошибки. Например, ALL MEN является допустимым исходным сообщением с делением на блоки AL LM EN, тогда как KISS ME и WHERE ARE YOU не удовлетворяют нашим правилам. Первый текст содержит в блоке деления две буквы S, а последний имеет нечетную длину.

2. Мы знаем, что каждый блок исходного сообщения содержит две различные буквы. Зашифрование блока осуществляется с помощью квадрата. Если две буквы не попадают в одну строку или столбец, к примеру A и E, то мы смотрим на буквы в углах прямоугольника, определяемого данной парой букв, в нашем случае — A, F, O, E. Пара AE отображается в FO. Порядок букв в паре FO определяется из условия, что F находится в той же строке, что и A, а O — в той же строке, что и E. Аналогично EA отображается в OF, OF в EA, SV в ZB, RC в IH, TL в ER. Если же две буквы попадают в одну и ту же строку (соответственно столбец), мы циклически смещаемся на один шаг вправо (соответственно вниз). Так HA переходит в CX, WX в UG, CA в AX, DM в PD и RL в IR.

Попытаемся теперь зашифровать текст CRYPTO ENIGMA. (Криптосистема, используемая немецкими вооруженными силами во второй мировой войне, базировалась на машине ENIGMA.) Деление на блоки исходного сообщения даст:

CR YP TO EN IG MA.

Заметим, что CR, YP и IG переходят в HI, DI и UN соответственно. Здесь мы действовали по правилу прямоугольника. Пары TO и EN лежат в одной строке и, значит, переходят в NG и TO соответственно. Наконец, пара MA лежит в одном столбце и переходит в DO. Таким образом получим криптотекст HIDING TO UNDO. Наш квадрат способен изумительно работать с семантикой!

Для квадрата Плейфейра не будет никаких различий, если некоторые столбцы переместить с одной стороны на другую, и строки — сверху вниз. Важно сохранить только циклический порядок строк и столбцов. Читатель может убедиться, что квадрат

P	U	L	R	I
A	X	F	H	C
O	G	E	T	N
M	Q	V	B	K
D	W	Z	S	Y

эквивалентен нашему исходному квадрату, так как они оба одинаково шифруют любой текст.

Наши правила для системы Плейфейра не означают, что возможны только они. Пары букв исходного сообщения могут трактоваться поразному, к примеру, вставкой специальной буквы (часто Q) в промежутки. Прямоугольник 5×5 может быть заменен на 4×4 или 3×9 с соответствующим изменением количества букв в алфавите. Так же пара, расположенная в одной строке (соответственно столбце), может быть зашифрована парой, находящейся циклически под искомой (соответственно справа).

Мы отмечали в параграфе 1.2, что главным объяснением для таких систем, как система Цезаря с ключевым словом, является управление ключом: вместо произвольной перестановки 26 букв мы имеем простой путь представления ключа. Такое простое представление возможно также и для системы Плейфейра. Вместо запоминания квадрата 5×5 из букв, мы хотим хранить в памяти что-нибудь более простое. Ключевые слова также могут использоваться и для системы Плейфейра. Мы выбираем ключевое слово без повтора букв. Начнем строить квадрат с ключевого слова, после которого расставим оставшиеся буквы (кроме J) в алфавитном порядке. Так, для ключевого слова HOW MANY ELKS получим следующий квадрат:

H	O	W	M	A
N	Y	E	L	K
S	B	C	D	F
G	I	P	Q	R
T	U	V	X	Z

Теперь мы опять готовы перейти в мир криптоаналитика. Сделаем это для большого примера.

Пример 1.5. Известный детектив Уайт исследовал дело о бесследно пропавшем мультимиллионере Ойле (J.R.Oil). С помощью гениальной дедукции, не имеющей к нам отношения, Уайт нашел зашифрованное письмо со следующим текстом:

QN	FS	LK	CM	LT	HC	SM	MC	VK
IH	HA	XR	QM	BQ	IE	QN	AK	RD
PS	TU	CB	NX	MC	IF	NX	MC	IT
YF	SD	EF	IF	QN	LQ	FL	YD	SB
QN	AK	EU	MC	TI	IE	QN	MS	IQ
KA	PF	IL	BM	WD	DF	RE	IV	KA
MC	IT	QN	FX	MB	FT	FT	DX	AK
HC	SM	YF	WE	BA	AB	QE	IV	OI
XT	IT	FM	AQ	AK	QN	MX	ZU	DS
OI	XI	QN	FY	RX	NV	OR	RB	RA
MC	MB	NX	XM	AE	OW	FT	LR	NC
IQ	QN	FM	ML	SN	AH	QN	QL	TW
FL	ST	LT	PI	QI	QN	DS	VK	AR
FS	AQ	TI	DF	SM	AK	FO	XM	VA
RZ	FT	SN	GS	UD	FM	SA	WA	LN
MF	IT	QN	FG	LN	BQ	QE	AR	VA
DT	FT	QA	AB	FY	IT	MX	DK	FM
DF	QN	FX	NO	XC	TF	SM	FK	OY
CM	QM	BA	LH					

Уайт пришел в сауну. Он знал, что жар сауны расширяет сосуды мозга, после чего очень хорошо думается. Согласно его опыту, наиболее трудными задачами были задачи “трех саун”, тогда как над этим делом он размышлял в течение одной сауны.

Вместе с зашифрованным письмом Уайт нашел прекрасно инкрустированный серебряный ключ. Его длина равнялась трем дюймам. Уайт знал Ойла как энтузиаста-спортсмена. Честная игра (Fair play) была одним из принципов, которого Ойл всегда придерживался. Так и есть! Система Плейфейра с ключом из трех букв! Уайт теперь был уверен, что расшифрует это письмо.

После возвращения из сауны Уайт нашел свои заметки о распределении пар букв — *диграмм*. В английском наиболее частыми парами, согласно [Ga], являются:

TH	6.3%	AR	2.0%	HA	1.7%
IN	3.1%	EN	2.0%	OU	1.4%
ER	2.7%	TI	2.0%	IT	1.4%
RE	2.5%	TE	1.9%	ES	1.4%
AN	2.2%	AT	1.8%	ST	1.4%
HE	2.2%	ON	1.7%	OR	1.4%

Хотя это не относится к настоящей задаче, Уайт также отметил наиболее часто встречающиеся пары в других языках.

Немецкий:	EN ER CH DE GE EI IE IN NE ND BE EL TE UN ST DI NO UE SE AU
Финский:	EN TA IS IN ST AN TT SI AA IT LL TE SE AI KA SA VA LI AL TI
Французский:	ES EN OU DE NT TE ON SE AI IT LE ET ME ER EM OI UN QU
Итальянский:	ER ES ON RE EL EN DE DI TI SI AL AN RA NT TA CO
Испанский:	ES EN EL DE LA OS AR UE RA RE ER AS ON ST AD AL OR TA CO

Уайт для себя отметил, что у него имеется также статистика о триграммах, тетраграммах и обратимых парах в различных языках. Он немного знал и о распределении гласных и согласных, о вероятности появления буквы в начале или конце слова.

Он учел, что PLAYFAIR уничтожает всю информацию о началах и концах слов, что некоторая информация будет утрачена, если считать пары только так, как они появляются в криптотексте, игнорируя пары, образованные разными парами, такие, как NF, SL, KC в начале текста. Однако он полностью осознавал, что статистики пар не могут быть абсолютными: некоторые статистики включают пары типа LM в CALL ME, в то время как другие не включают их. Поэтому Уайт сделал вывод, что информации, извлекаемой из подсчета частоты появления пар в криптотексте, вполне достаточно для криптоанализа.

Имеется 97 различных пар среди 166 пар криптотекста. 97 составляет 16,2% от всех возможных $25 \cdot 24 = 600$ пар в системе Плейфейра. Уайт знал, что это вполне нормально: даже в более длинном тексте маловероятно, что вы используете более 40% всех возможных пар. Большинство из теоретически возможных пар никогда не появляются в английском.

Пары, появляющиеся в криптотексте более трех раз:

QN,	13 появлений,	7.8%
MC,	6 появлений,	3.6%
AK,	5 появлений,	3.0%
FT,	5 появлений,	3.0%
IT,	5 появлений,	3.0%
FM,	4 появления,	2.4%
SM,	4 появления,	2.4%

Уайт знал, что это предварительная информация. Он изучил также другие пары, например буквы, составляющие пары с многими буквами. Однако он хотел приступить к направленной атаке. Очевидно, что QN

есть замаскированная пара TH. Какую информацию можно извлечь из этого?

На рис. 1.4 изображен квадрат Уайта для системы Плейфейра. Длина ключевого слова равна трем. После ключа все буквы следуют в алфавитном порядке.

Рис. 1.4.

Таким образом, TH отображается в QN. Это невозможно, если H, N, Q, T расположены в одной строке, так как алфавитный порядок не будет сохранен. Что можно сказать об их расположении в одном столбце? T циклически предшествует Q, и H циклически предшествует N. Согласно алфавитному порядку T должна стоять в нижней строке, а Q — в верхней строке. Однако буквы U, V, W, X, Y, Z следуют за T, за исключением букв, появляющихся в ключевом слове. Это возможно только в том случае, когда две из шести данных букв входят в ключевое слово и T лежит в самом левом столбце. Это означает, что квадрат имеет вид

Q	U	X	A	B
C	D	E	F	G
H	I	K	L	M
N	O	P	R	S
T	V	W	Y	Z

Возможны лишь такие варианты, когда вместо U и X после Q в ключевом слове могут появляться любые две буквы из U, V, W, X, Y, Z. Оставшиеся 4 буквы тогда следуют за T в нижней строке в алфавитном порядке.

Имеет ли это какой-нибудь смысл? Уайт отметил, рассматривая частоты других пар, что MC получается из HG, FM из GL, а SM из MG.

AK, FT и IT получены из очень редких, можно сказать несуществующих, английских пар. Уайт заключил, что квадрат построен неправильно и, следовательно, пара QN должна получаться из TH по правилу прямоугольника.

Этот прямоугольник должен лежать в квадрате после ключевого слова. Иначе невозможно сохранить алфавитный порядок. Следовательно, прямоугольник выглядит так:

$$\begin{array}{ccc} \mathrm{H} & \ldots & \mathrm{N} \\ & \vdots & \\ \mathrm{Q} & \ldots & \mathrm{T} \end{array}$$

Буквы I, K, L, M должны быть расположены между H и N, буквы O, P — между N и Q, буквы R, S — между Q и T. При этом имеется в виду, что не более трех из данных промежуточных букв могут отсутствовать, так как они могут появиться в ключевом слове. Конечно, в силу алфавитного порядка нет других букв кроме этих, которые могли бы располагаться между данными тремя парами.

Кроме того, H, N, Q, T должны образовывать прямоугольник. Сколько букв из I, K, L, M входят в ключевое слово? Менее двух не может быть, потому что между Q и T не более двух букв. Более двух букв также невозможно, так как в этом случае будет много букв в ключе. Следовательно, *в точности две буквы из* I, K, L, M *входят в ключевое слово*. Поэтому *в точности одна буква из* O, P *входит в ключевое слово*. Иначе не будет образован прямоугольник.

Каким может быть ключевое слово? Зная Ойла, ответ был очевиден для Уайта: ключевое слово — OIL! Уайт быстро построил квадрат:

$$\begin{array}{ccccc} \mathrm{O} & \mathrm{I} & \mathrm{L} & \mathrm{A} & \mathrm{B} \\ \mathrm{C} & \mathrm{D} & \mathrm{E} & \mathrm{F} & \mathrm{G} \\ \mathrm{H} & \mathrm{K} & \mathrm{M} & \mathrm{N} & \mathrm{P} \\ \mathrm{Q} & \mathrm{R} & \mathrm{S} & \mathrm{T} & \mathrm{U} \\ \mathrm{V} & \mathrm{W} & \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{array}$$

и начал расшифрование:

TH	ET	IM	EH	AS	CO	ME	HE	WH
OK	NO	WS	SH	OU	LD	TH	IN	KI
MU	ST	GO	MY	HE	AD	MY	HE	AR
TA	RE	DE	AD	TH	OS	EA	WF	UL
TH	IN	GS	HE	RA	LD	TH	EM	OR
NI	NG	OI	LP	RI	CE	SD	OW	NI
HE	AR	TH	EY	PL	AN	AN	EW	IN
CO	ME	TA	XD	AL	LA	SC	OW	BO
YS	AR	EN	OT	IN	TH	ES	UP	ER
BO	WL	TH	AT	SW	HY	IQ	UI	TI
HE	LP	MY	SE	LF	IV	AN	IS	HF
OR	TH	EN	EX	TM	ON	TH	SO	RY
EA	RS	AS	KB	RO	TH	ER	WH	IT
ET	OT	RA	CE	ME	IN	CA	SE	YO
UW	AN	TM	EU	RG	EN	TL	YI	AM
NE	AR	TH	EF	AM	OU	SC	IT	YO
FR	AN	TO	LA	AT	AR	ES	ID	EN
CE	TH	EY	HA	VE	NA	ME	DN	AV
EH	SH	AL	OM					

Уайт переписал это в нормальном виде, учитывая знаки препинания:

The time has come. He who knows should think. I must go. My head, my heart are dead. Those awful things herald the morning. Oil prices down. I heat they plan a new income tax. Dallas Cowboys are not in the Superbowl. That's why I quit. I help myself. I vanish for the next months or years. Ask Brother White to trace me in case you want me urgently. I am near the famous city of Rantola at a residence they have named Naveh Shalom.

Уайт знал, что ему повезло с догадками. Однако также были верны и его рассуждения, основанные на правильных предположениях. Хорошему криптоаналитику, как хорошему голкиперу, должна сопутствовать и удача. Рассматриваемое дело было закрыто.

□

Напомним главную идею *многоалфавитных* систем. Первая буква исходного сообщения шифруется как обычно, в то время как следующая буква может шифроваться по другому принципу. Так, буква A может быть зашифрована многими способами; замены для A и других букв выбираются из *многих алфавитов*. Это также хорошая защита против простого подсчета частот: не существует единой маскировки для буквы A в криптотексте.

Одной из старейших и наиболее известных многоалфавитных систем является система Виженера, названная в честь французского криптографа Блейза Виженера (1523–1596).

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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

Рис. 1.5.

Система Виженера подобна системе Цезаря, в которой ключ меняется от шага к шагу. Квадрат Виженера на рис. 1.5 обычно используется для зашифрования и расшифрования. Каждый столбец может быть рассмотрен как система Цезаря с ключами 0, 1,. . . , 25. Для зашифрования читаем исходное сообщение из строк и ключи системы Цезаря из столбцов. Ключи обычно выражаются в терминах *ключевого слова.* К примеру, для зашифрования исходного сообщения PURPLE под ключевым словом CRYPTO мы сначала находим пересечение P-строки и C-столбца и

получаем R. Весь криптотекст выглядит как RLPEES. Криптотекст будет тем же самым, если мы поменяем роли строк и столбцов в процессе шифровки. Для расшифрования находим, в какой строке в C-столбце лежит R. Таким способом получаем P и т. д.

Ключевое слово обычно применяется *периодически*. Если сообщение длиннее периода, то ключевое слово повторяется сначала. К примеру, ключевое слово CRYPTO применяется к сообщению из 15 букв в виде CRYPTOCRYPTOCRY.

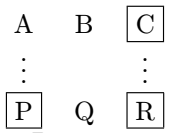
```
Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
A Z Y X W V U T S R Q P O N M L K J I H G F E D C B
B A Z Y X W V U T S R Q P O N M L K J I H G F E D C
C B A Z Y X W V U T S R Q P O N M L K J I H G F E D
D C B A Z Y X W V U T S R Q P O N M L K J I H G F E
E D C B A Z Y X W V U T S R Q P O N M L K J I H G F
F E D C B A Z Y X W V U T S R Q P O N M L K J I H G
G F E D C B A Z Y X W V U T S R Q P O N M L K J I H
H G F E D C B A Z Y X W V U T S R Q P O N M L K J I
I H G F E D C B A Z Y X W V U T S R Q P O N M L K J
J I H G F E D C B A Z Y X W V U T S R Q P O N M L K
K J I H G F E D C B A Z Y X W V U T S R Q P O N M L
L K J I H G F E D C B A Z Y X W V U T S R Q P O N M
M L K J I H G F E D C B A Z Y X W V U T S R Q P O N
N M L K J I H G F E D C B A Z Y X W V U T S R Q P O
O N M L K J I H G F E D C B A Z Y X W V U T S R Q P
P O N M L K J I H G F E D C B A Z Y X W V U T S R Q
R Q P O N M L K J I H G F E D C B A Z Y X W V U T S
S R Q P O N M L K J I H G F E D C B A Z Y X W V U T
T S R Q P O N M L K J I H G F E D C B A Z Y X W V U
U T S R Q P O N M L K J I H G F E D C B A Z Y X W V
V U T S R Q P O N M L K J I H G F E D C B A Z Y X W
W V U T S R Q P O N M L K J I H G F E D C B A Z Y X
Y X W V U T S R Q P O N M L K J I H G F E D C B A Z
```

Рис. 1.6. Квадрат Бьюфорта.

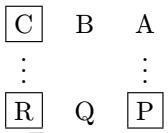
Существует, конечно, много других легкозапоминающихся квадратов, которые могут применяться в качестве основы для многоалфавитной системы так же, как и квадрат Виженера. Одним из наиболее известных является *квадрат Бьюфорта* на рис. 1.6: его строками являются строки квадрата Виженера, записанные в обратном порядке. Он

назван в честь адмирала сэра Фрэнсиса Бьюфорта — создателя шкалы для определения скорости ветра.

Если в квадрате Виженера первая строка и столбец указывают на строки и столбцы соответственно, то в квадрате Бьюфорта этим целям служат первая строка и последний столбец. Так, первая буква криптотекста при зашифровании сообщения CRYPTO получается из двух квадратов следующим образом:



Виженер



Бьюфорт

Термин *периодический* относится к многоалфавитным системам, в которых алфавиты подстановок периодически повторяются. Типичным примером является система Виженера с периодически повторяющимся ключевым словом, как описано выше. Если известен период, то криптоанализ данной системы может быть сведен к криптоанализу одноалфавитных систем, как показано ниже. Пусть период равен пяти. Упорядочим буквы криптотекста в пяти столбцах следующим образом. Число указывает позицию буквы в криптотексте:

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
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Два появления одинаковой буквы в одном столбце представляют одну букву сообщения. Поэтому можно расшифровать каждый столбец простым подсчетом частот.

Примерно в 1860 г. немецким криптоаналитиком Ф.У. Казизки был изобретен метод для вскрытия периодических криптосистем с неизвестным периодом. *Метод Казизки* выявляет период с помощью обнаружения одинаковых слов в криптотексте. Скажем, слово PUXUL появляется дважды, с 15 буквами между двумя появлениями:

... PUXUL 15 букв PUXUL.

Это может быть чисто случайно. А может означать тот факт, что одинаковая часть сообщения зашифрована, начиная с той же позиции ключа. Тогда расстояние между двумя P равно 20 и кратно длине ключа. Поэтому возможная длина ключа есть 2, 4, 5, 10 или 20. Когда формируется несколько таких предположений о длине ключа — некоторые из предположений будут, возможно, неправильными, — может быть сделано верное предположение о длине ключа. Более длинные повторяющиеся слова предпочтительнее. Также является преимуществом для криптоаналитика повторение слов более одного раза.

Метод Казизки проиллюстрируем на следующем примере.

Пример 1.6. Криптоаналитик, подозревая, что используется система Виженера, перехватил следующий криптотекст.

A V X Z H H C S B Z H A L V X H F M V T L H I G H
K A L B R V I M O F H D K T A S K V B M O S L A C
G L G M O S T P F U L Q H T S L T C K L V N T W W
H B W M S X S G A V H M L F R V I T Y S M O I L H
P E L H H L L I L F B L B V L P H A V W Y M T U R
A B A B K V X H H B U G T B B T A V X H F M V T L
H I G H P N P Z W P B Z P G G V H W P G V B G L L
R A L F X A V X T C L A Q H T A H U A B Z H T R S
B U P N P Z W P B Z H G T B B T P G M V V T C S M
V C L T O E S O L A C O L K B A V M V C Y L K L A
C G L G B M H A L G M V J X P G H U Z R H A B Z S
K H P E L H B U M F L H T S P H E K B A V T J C N
W Z X V T L A C G L G H U H H W H A L B M O S K V
C F J O G U C M I S A L O M L R I Y C I L F E F I
G S S L Z W M P G O L F R Z A T S Z G L J X Y P X
Z H B U U R D W M O H A L V X H F M V T L H I G H

При этом нет известных пар открытый текст-криптотекст. Криптоаналитик может разбить данный криптотекст, содержащий ровно 400 букв, скажем, на блоки из 5 букв. Однако он не учитывает деление на блоки и использует метод Казизки. Деление на блоки при этом доставляет только неприятности, так как одинаковые слова могут появляться, начиная с любой позиции в блоке.

Криптоаналитик замечает, что слово

H A L V X H F M V T L H I G H ,

необычно большое по отношению к длине всего текста, появляется дважды. Расстояние между двумя этими появлениями равно $375 = 3 \cdot 5^3$ и

вычисляется с помощью фиксации некоторой буквы, скажем H, встречающейся в обоих появлениях, путем подсчета числа шагов от ее первого появления до выделенного второго. В данном случае это легко, так как очевидно, что число шагов равно $15 \cdot 25$.

Конечная часть рассматриваемого слова, а именно,

V X H F M V T L H I G H

появляется также и в третий раз. Расстояние между первыми двумя появлениями равно $129 = 3 \cdot 43$, а расстояние между следующими двумя появлениями равно $246 = 2 \cdot 3 \cdot 41$.

Общим делителем всех этих чисел является только 3. Так как рассматриваемые слова являются достаточно длинными, то криптоаналитик догадывается, что их появление не является случайным. Наоборот, предполагается, что одинаковая последовательность букв зашифрована, начиная с одинаковой позиции в ключевом слове. Если их появление не случайно, то период обязательно равен 3.

Криптоаналитик, используя компьютер, очень легко проведет исчерпывающий поиск всех повторяющихся слов длины по крайней мере 2. Кроме этого, он пытается проводить поиск, делая ставку на период 3. Пара незамедлительных наблюдений поддерживает это решение. Имеется другое появление VXH через 12 шагов. Также есть три появления AVX с расстояниями 141 и 39 друг от друга. Существует 4 появления HAL с расстояниями 246, 60 и 69 друг от друга. Все эти числа делятся на 3, в то время как другой делитель приводил бы к периоду, не согласующемуся с остальной информацией.

Криптоаналитик знает, что такая направленная атака, уклоняясь от случайного поиска, будет законной и с теоретической точки зрения. В простом примере, как у нас, направленная атака может быть проведена и без помощи компьютера: криптоаналитик может сделать все вручную. Более важно, что в запутанных примерах из “реальной жизни” подобная атака помогает свести задачу от неподдающейся обработке к вполне разрешимой.

Предполагая, что период равен трем, путем простого подсчета частот получим следующее распределение букв по трем классам. Буквы в классе 1 имеют позиции 1, 4, 7, ...

Буква	Класс 1	Класс 2	Класс 3
A	12 = 9.0 %	5	9 = 6.8 %
B	4	9 = 6.8 %	12 = 9.0%
C	2	11 = 8.3 %	-
D	2	-	-
E	1	-	4

Буква	Класс 1	Класс 2	Класс 3
F	1	10 = 7.5 %	2
G	-	13 = 9.8 %	10 = 7.5 %
H	15 = 11.2 %	14 = 10.5 %	11 = 8.3 %
I	1	7	3
J	2	2	-
K	1	5	4
L	27 = 20.1 %	1	13 = 9.8 %
M	2	2	17 = 12.8 %
N	-	-	4
O	6	4	2
P	10 = 7.5 %	7	-
Q	-	2	-
R	1	3	5
S	5	13 = 9.8 %	-
T	6	4	13 = 9.8 %
U	9 = 6.7 %	1	1
V	14 = 10.4 %	11 = 8.3 %	2
W	2	3	6
X	-	1	12 = 9.0 %
Y	4	-	1
Z	7	5	2

R S T — это три последовательные буквы из высокочастотной группы ETAONISRH. Поэтому криптоаналитик ищет в каждом из трех классов по три последовательных буквы с высокой частотой появления для каждой. Таким способом он находит, как шифруются R S и T в каждом классе.

В классе 1 имеются две последовательности высокочастотных букв: TUV и YZA. Если TUV представляет RST, то сдвиг равен двум, но тогда буквы сообщения WXY имеют высокие частоты 4, 7, 12. Поэтому для представления RST выбираем YZA, и сдвиг в данном случае равен 7. Это означает, что самая высокочастотная буква L (20.1%) шифрует E. В небольших примерах (в нашем случае только 134 буквы) мы не можем быть уверены, что буква с наивысшей частотой действительно соответствует E. Тем не менее, как и здесь, обычно может быть выбрана только E.

В классе 2 криптоаналитик делает аналогичный выбор между ABC и FGH (также рассматриваются ZAB и GHI). По той же причине,что и ранее, выбираем FGH, что дает сдвиг 14. В классе 3 существует только один выбор — KLM, дающий сдвиг 19. Заметим, что ни в классе 2, ни в классе 3 буква E не является самой высокочастотной, хотя входит в

такого типа группы в обоих классах.

Три сдвига 7, 14, 19 получаются из ключевого слова HOT. Криптоаналитик может начать расшифровку:

T H E S T O V E I S T H E H E A R T O F S A U N A
W H E N Y O U T H R O W W A T E R O N T H E S T O
N E S T H E A I R B E C O M E S M O R E H U M I D

Работа завершена: сообщение содержит информацию о сауне. Теперь криптоаналитик переписывает сообщение, используя знаки пунктуации.

The stove is the heart of sauna. When you throw water on the stones, the air becomes more humid and feels hotter. You are, thus, able to experience both dry and humid heat in sauna. The art of sauna building is not discussed here. The most common mistake in building a sauna is to have too small a stove with too few stones. If the stove is only a miserable tiny metal box with a couple of stones on top, then the room cannot be heated properly unless it is very small. Never be stingy with the heart of sauna!

Криптоаналитик еще раз посмотрит на свою работу. Факты, использованные в качестве основы для анализа методом Казизки, оказались правильными: одинаковые последовательности букв шифровались с одинаковой позиции в периоде. Слова AVX и HAL являются двумя шифровками сообщения THE, начиная с первой и второй позиций периода соответственно. Иногда идентичные части сообщения, одинаково зашифрованные, имеют совершенно разные синтаксические и/или семантические функции. Так, VXH был шифровкой для HEA. Но HEA приходит из слов HEART, HEATING, также как и THE ART.

Несмотря на небольшой размер классов, высокочастотные буквы в каждом классе почти всегда совпадали с ETAONISRH. В действительности каждая “реально высокочастотная” буква (т.е. буква с не менее 9 появлениями и процентами, указанными выше) попадала в эту группу.

Заключительным выводом криптоаналитика являлось то, что период должен быть намного больше длины рассматриваемого сообщения.

□

В наших криптоаналитических примерах использовались некоторые известные свойства естественных языков: частота индивидуальных букв и частота пар букв (диграмм). Мы хотим подчеркнуть, что

допустимы статистики и о многих других свойствах, к примеру, частота триграмм, широко распространенные слова языка, наиболее вероятные левый и правый соседи каждой буквы, распределение и совместные сочетания гласных и согласных. Во многих криптоаналитических задачах такие дополнительные статистики крайне удачны для исключения большинства из возможных альтернатив.

Дальнейшей модификацией системы Виженера является система AUTOCLAVE, приписываемая математику XVI в. Дж. Кардано, который известен своими формулами для решения уравнений третьей и четвертой степеней. В системе AUTOCLAVE сообщение служит также с определенным сдвигом и ключом шифровки. В следующем примере сдвиг равен 6.

Сообщение: A I D S I S T R A N S M I T T E D T H R O U G H
Ключ: A I D S I S T R A N S M I T T E D T

Ключ используется, как и в системе Виженера, для определения подстановки Цезаря для каждой буквы. Пустое пространство в начале ключа может быть заполнено циклически концом сообщения или использованием *ключевого слова.* Ключевое слово IMMUNE дает следующее начало для криптотекста:

Сообщение: A I D S I S T R A N S M I T T E D T H R O U G H
Ключ: I M M U N E A I D S I S T R A N S M I T T E D T
Криптотекст: I U P M V W T Z D F A E B K T R V F P K H Y J A

Легальная расшифровка очевидна: ключевое слово позволяет получить начало сообщения из начала криптотекста, после чего найденная часть исходного сообщения используется в качестве ключа.

В другом варианте системы AUTOCLAVE в качестве ключа служит криптотекст, записанный после ключевого слова. В данном случае наш предыдущий пример будет зашифрован следующим образом:

Сообщение: A I D S I S T R A N S M I T T E D T H R O U G H
Ключ: I M M U N E I U P M V W B L P Z N I J E I D O B
Криптотекст: I U P M V W B L P Z N I J E I D Q B Q V W X W I

Криптоанализ последней версии системы AUTOCLAVE будет более успешен: аналитику достаточно только угадать или найти *длину* ключа. Предположим, известно, что для примера, указанного выше, длина ключа равна 6. Тогда аналитик берет первую букву I и седьмую букву B из криптотекста. Буква B лежит в T-строке и I-столбце в квадрате Виженера. Это дает букву T исходного сообщения. Аналогично буква R исходного сообщения получается из U и L. Подобным

образом может быть раскрыт весь исходный текст, кроме первых шести букв. Первая версия системы AUTOCLAVE (где ключом служит сдвиг исходного сообщения) неуязвима против такой простой криптоаналитической атаки.

Теперь мы коротко набросаем в общих чертах криптоанализ первой версии системы AUTOCLAVE. Метод Казизки требует нахождения длины ключевого слова или, по крайней мере, некоторых предположений о длине, являющейся здесь также и *периодом*. Теоретическое обоснование метода Казизки не так удачно здесь, как для системы Виженера, но метод обычно хорош даже для нахождения периода. Рассмотрим пример. Пусть слово THE появляется дважды в исходном сообщении и расстояние между этими появлениями равно удвоенному периоду. Тогда найдется некоторая последовательность из трех букв, скажем AID, расположенная посередине между появлениями THE. Таким образом, имеется следующая часть исходного сообщения:

$$\ldots \text{THE} \ldots \text{AID} \ldots \text{THE} \ldots$$

В процессе зашифрования получим

Сообщение:	$\ldots$	THE	$\ldots$	AID	$\ldots$	THE	$\ldots$
Ключ:			$\ldots$	THE	$\ldots$	AID	$\ldots$
Криптотекст:			$\ldots$	TPH	$\ldots$	TPH	$\ldots$

Таким образом, TPH появляется дважды в криптотексте и расстояние между данными появлениями будет равно периоду. Метод Казизки дает здесь *в точности период*, в то время как для системы Виженера он давал лишь числа, *кратные периоду*.

После того как период известен, скажем, что он, вероятно, равен 6, ключевое слово находится с помощью исчерпывающего поиска, основанного на подсчете частот индивидуальных букв. Когда известно ключевое слово, все, конечно, становится очевидным.

Имеется 26 возможных выборов для первой буквы ключевого слова. Когда этот выбор зафиксирован, из него определяется вместе с первой буквой криптотекста первая буква сообщения. Последняя, в обратном порядке, определяет вместе с седьмой буквой криптотекста седьмую букву сообщения. И так далее. Поэтому каждый выбор первой буквы ключевого слова дает нам буквы исходного сообщения в позициях 1, 7, 13, 19, 25,.... Варианты, приводящие к последовательностям невозможных распределений, могут быть отброшены. Таким способом находится первая буква. Остальные 5 букв обнаруживаются аналогично.

Мы обсудили основные криптоаналитические методы для наиболее известных старых криптосистем. Сделаем несколько дополнительных замечаний. Не существует универсального метода, который может

быть рекомендован для всех криптоаналитических задач. Поэтому криптоаналитик всегда должен быть активен: если один метод не принес успеха, необходимо пытаться применить другой.

Сообщение почти всегда является текстом естественного языка, при этом допускается наличие некоторого промежуточного кодирования. Криптоаналитик наверняка знает, какой из языков используется для связи. Очень часто это можно понять из "истории перехвата" криптотекста, но мы также не должны забывать Золотое правило для проектирования криптосистем! Криптоаналитик должен знать язык исходного сообщения или по крайней мере сотрудничать со знающим его лицом. Следовательно, повысить секретность можно с помощью использования в качестве языка исходных сообщений малораспространенного языка, например такого, как финский. Здесь подходящее место для открытия алгоритма зашифрования, используемого в примере 1.1. Сообщением было WEMEETTOMORROW. Вначале оно было переведено на финский: TAPAAMMEHUOMENNA. Используя затем алгоритм зашифрования E_1 системы Цезаря (сдвиг на один шаг), получаем криптотекст UBQBBNNFIVPNFOOB.

Мы обсудили различия между одноалфавитными и многоалфавитными системами. Другая естественная классификация, основанная на теории формальных языков, делит криптосистемы на *контекстносвободные* и *контекстнозависимые*. В первых шифруются индивидуальные буквы, а в последних — группы букв. Это может происходить как в одноалфавитных, так и многоалфавитных системах. Типичные примеры криптосистем различных типов даны в следующей таблице:

Системы	Контекстносвободные	Контекстнозависимые
Одноалфавитные	Цезаря	Плейфейра
Многоалфавитные	Виженера	Плейфейра с периодом

Здесь система Плейфейра с периодом означает модификацию системы Плейфейра, где вместо одного используются несколько квадратов, скажем три. Первая пара сообщения шифруется согласно первому квадрату, вторая и третья пары согласно второму и третьему квадратам, четвертая пара опять согласно первому квадрату и т. д.

Подводя итог этого параграфа, отметим еще несколько криптосистем совершенно различной природы. Система "кодовая книга" (CODE BOOK) описана в [Ca] как аристократ среди всех криптосистем. В этом утверждении есть доля истины, так как в кодовой книге учитывается много аспектов, делающих криптотекст не вызывающим подозрений.

Обе легальные части имеют словарь перевода слов исходного сообщения (по крайней мере, наиболее необходимых) в последовательности

чисел, некоторые бессмысленные слова или, что предпочтительнее, в некоторые другие осмысленные слова. При этом часть словаря может выглядеть следующим образом:

Оригинал	Перевод
ATTACK	FISHING
⋮	⋮
IN	BETWEEN
⋮	⋮
MORNING	WORK HOUR
⋮	⋮
THE	THE

Теперь сообщение ATTACK IN THE MORNING (атака утром) шифруется в криптотекст FISHING BETWEEN THE WORK HOURS (рыбалка в рабочем перерыве). Чтобы сделать криптотекст синтаксически правильным, в него добавляются разумные концовки.

Что можно сказать о криптоанализе кодовой книги? Если ничего не известно о словаре, то начальное условие “известен только криптотекст” ничего не дает. С другой стороны, начальные условия “известно некоторое сообщение” и “известно избранное сообщение” обязательно приоткрывают некоторые детали словаря. Насколько сильно это может помочь при криптоанализе, зависит от деталей.

Существуют ли криптосистемы, которые гарантируют идеальную секретность? Безупречная секретность, кратко говоря, означает, что криптотекст не дает никакой информации для криптоаналитика. Криптоаналитик, перехватывает ли он криптотексты или нет, всегда имеет одинаковые знания. Криптотекст не дает информации об исходном сообщении.

Примером криптосистемы с идеальной секретностью является “одноразовый блокнот” (ONE-TIME PAD). Сообщением является последовательность битов ограниченной длины, скажем последовательность, содержащая не более 20 бит. В виде ключа выступает последовательность из 20 бит. Ключ используется одновременно для зашифрования и расшифрования и передается получателю через некоторый секретный канал. Возьмем ключ 11010100001100010010.

Сообщение, скажем 010001101011, шифруется, используя побитовое сложение с ключом, начиная с его начала. Таким образом, криптотекстом является 100100101000. Это не дает информации для криптоаналитика, потому что он не знает, появился ли бит криптотекста прямо

из исходного сообщения или изменился под действием ключа. Здесь существенным является одноразовое использование ключа, на что указывает название криптосистемы. Предыдущее сообщение вместе с соответствующим криптотекстом дает ключ или, по крайней мере, префикс ключа. Также предоставляет некоторую информацию и набор предыдущих нерасшифрованных криптотекстов. Конечно, при легальном расшифровании, очевидно, используется побитовое сложение криптотекста с началом ключа.

Очевидным недостатком одноразового блокнота является сложное управление ключом. Ключ, с длиной не меньшей длины исходного сообщения, должен передаваться отдельно через некоторый секретный канал. Нет никакого выигрыша: трудности в секретной связи только переходят на другой уровень! Конечно, система еще может использоваться для передачи действительно важных одноразовых сообщений.

В некоторых вариантах одноразового блокнота управление ключом является легким, но секретность не равна в точности 100% . Обратим внимание на следующий пример.

Ключ определяется указанием места в Библии (версия короля Джеймса). Для примера, Джошуа 3, 2, 6 означает книгу Джошуа, часть 3, абзац 2, письмо 6. Ключ начинается с этого письма и используется таким же образом, как в системе Виженера. Зашифруем сообщение PRACTICAL PERFECTLY SECRET SYSTEMS WOULD CAUSE UNEMPLOYMENT AMONG CRYPTOGRAPHERS[3], используя этот ключ.

Сообщение:	P R A C T I C A L P E R F E C T L Y S E C R E T
Ключ:	C A M E T O P A S S A F T E R T H R E E D A Y S
Криптотекст:	R R M G M W R A D H E W Y I T M S P W I F R C L

Сообщение:	S Y S T E M S W O U L D C A U S E U N E M P L O
Ключ:	T H A T T H E O F F I C E R S W E N T T H R O U
Криптотекст:	L F S M X T W K T Z T F G R M O I H G X T G Z I

Сообщение:	Y M E N T A M O N G C R Y P T O G R A P H E R S
Ключ:	G H T H E H O S T A T D T H E Y C O M M A N D E
Криптотекст:	E T X U X H A G G G R U R W X M I F M B H R U W

[3]Практические идеально секретные системы породят безработицу среди криптографов.

Управление ключом в этом варианте одноразового блокнота намного легче, так как очень длинные ключи могут быть представлены в компактной форме. Но, с другой стороны, ключи не будут случайными. Поэтому может быть использована информация о частотах, касающаяся английского языка. Возможен также и полный компьютерный перебор всех ключей.

1.4. Роторы и DES

Рассмотренные криптосистемы могут быть сделаны более хитроумными и в то же время более секретными с помощью использования криптографических машин. Такие машины осуществляют процессы зашифрования и расшифрования намного быстрее, а также обеспечивают возможность выбора ключей из потенциально неограниченного множества.

История криптографических машин насчитывает уже несколько сотен лет. Если ранние механические устройства затрачивали несколько секунд для шифровки символа, то современные электронные машины шифруют миллионы символов в секунду.

В этом заключительном параграфе, посвященном классической криптографии, мы обсудим некоторые из основ криптографических машин. Основная идея уже проявляется в старейшей машине, изобретенной и используемой Томасом Джефферсоном, — *колесе Джефферсона*.

Читатель может найти в [Ka] оригинальное описание колеса в терминологии самого Джефферсона.

Колесо Джефферсона состоит из цилиндра, надетого на ось. На цилиндре на одинаковом расстоянии друг от друга проведено 26 прямых линий, параллельных оси. Затем цилиндр разрезается на 10 меньших цилиндров одинаковой высоты. Эти меньшие цилиндры называются *дисками*. Таким образом, мы имеем 10 дисков, свободно вращающихся вокруг общей оси независимо друг от друга. Далее каждый из дисков делится по его окружности на 26 ячеек одинакового размера. Для каждого диска теперь сопоставим 26 ячейкам 26 букв английского алфавита. Порядок букв выбирается произвольно и меняется от диска к диску.

Конкретное колесо Джефферсона изображено на рис. 1.7. Оно же используется в примере 1.7, где индивидуальные диски будут описаны более детально, как и невидимые на рисунке части.

Добавим, что Джефферсон использовал 36 дисков. Мы выбираем меньшее число 10 для простоты изложения.

И отправитель и получатель имеют одинаковые колеса, т.е. циклические порядки букв одинаковы на каждом диске. Шифруя сообщение на английском, отправитель сначала делит его на блоки по 10 букв в каждом. Блок шифруется с помощью поворота дисков, чтобы он совпадал с одной из 26 буквенных последовательностей, параллельных оси, а в качестве криптотекста выбирается любая из оставшихся 25 буквенных последовательностей.

Рис. 1.7.

При расшифровании легальный получатель вращает диски колеса Джефферсона таким образом, чтобы криптотекст являлся одной из 26 буквенных последовательностей. Тогда исходное сообщение совпадает с одной из оставшихся 25 буквенных последовательностей. Так как с высокой вероятностью только одна из таких последовательностей букв может быть частью осмысленного английского текста, то выбор очевиден. Таким образом, необязательно договариваться заранее, сколько линий колеса будет использоваться в процессе шифрования. Это может быть любое число от 1 до 25, и оно может меняться от блока к блоку.

Ситуация несколько изменяется, когда сообщение не имеет смысла. В этом случае необходимо заранее договориться о сдвиге шифровки в колесе. К примеру, если сдвиг шифровки равен трем, то сообщение AAAAAAAAAA будет зашифровано как ESYMTRHUEE, согласно колесу на рис. 1.7.

Пример 1.7. Рассмотрим вновь колесо, изображенное на рис. 1.7, однако теперь мы опишем каждый из дисков, определив все последовательности букв. Такая процедура может быть проделана для определения любого колеса Джефферсона.

Номер диска:	1	2	3	4	5	6	7	8	9	10
Номер строки:										
1	A	A	A	A	A	A	A	A	A	A
2	R	R	P	N	V	S	P	E	I	I

Номер диска:	1	2	3	4	5	6	7	8	9	10
Номер строки:										
3	I	O	S	I	O	O	U	S	R	H
4	E	S	Y	M	T	R	H	U	E	E
5	K	U	L	O	Y	P	I	P	S	T
6	O	V	U	C	L	M	S	B	L	O
7	B	I	K	U	E	U	E	L	B	M
8	C	J	B	L	B	B	N	C	C	U
9	U	L	R	T	C	D	R	D	D	C
10	D	B	C	Y	D	Y	Y	H	F	D
11	J	F	D	B	G	E	D	I	N	F
12	T	C	T	F	F	C	B	J	Y	G
13	L	G	F	G	K	V	F	F	T	J
14	N	K	G	S	N	H	G	O	G	P
15	P	N	O	H	H	F	V	G	H	Q
16	W	P	N	J	U	K	J	K	J	B
17	Q	Q	E	D	P	L	K	M	K	N
18	M	T	H	E	Q	Q	M	N	M	V
19	S	H	M	K	R	I	T	Q	P	W
20	V	E	Q	P	S	J	O	R	Q	X
21	X	D	V	Q	W	N	L	V	V	L
22	Z	Y	W	V	X	G	W	W	W	Y
23	G	W	X	X	M	T	Q	Y	O	K
24	H	X	Z	R	I	W	X	X	U	R
25	Y	Z	I	Z	J	X	Z	T	X	S
26	F	M	J	W	Z	Z	C	Z	Z	Z

Оказывается, данное колесо Джефферсона имеет замечательные свойства для определенных сообщений.

Рассмотрим следующее сообщение. Оно содержит несколько вопросов о сауне. Сообщение состоит из 70 букв. Разделим его на блоки по 10 букв в каждом:

W H A T I S T H E B E S T T E M P E R A
T U R E I N S A U N A H O W M A N Y T I
M E S M U S T O N E G O I N H O W L O N
G M U S T L S T A Y

Отправитель решает использовать расстояния

8, 5, 6, 2, 13, 4, 3

для зашифрования данных семи блоков. Не имея колеса под руками, будем вращать диски мысленно. Для каждого из семи блоков мы вращаем диски таким образом, что блок может быть прочитан из строки 1.

Для данных семи случаев колесо имеет следующий вид. (Далее мы укажем только те строки, которые лежат на расстоянии, не превышающем выбранного сдвига.)

Номер диска:	1	2	3	4	5	6	7	8	9	10
Блок 1	W	H	A	T	I	S	T	H	E	B
	Q	E	P	Y	J	O	O	I	S	N
	M	D	S	B	Z	R	L	J	L	V
	S	Y	Y	F	A	P	W	F	B	W
	V	W	L	G	V	M	Q	O	C	X
	X	X	U	S	O	U	X	G	D	L
	Z	Z	K	H	T	B	Z	K	F	Y
	G	M	B	J	Y	D	C	M	N	K
	H	A	R	D	L	Y	A	N	Y	R
Блок 2	E	S	T	T	E	M	P	E	R	A
	K	U	F	Y	B	U	U	S	E	I
	O	V	G	B	C	B	H	U	S	H
	B	I	O	F	D	D	I	P	L	E
	C	J	N	G	G	Y	S	B	B	T
	U	L	E	S	F	E	E	L	C	O
Блок 3	T	U	R	E	I	N	S	A	U	N
	L	V	C	K	J	G	E	E	X	V
	N	I	D	P	Z	T	N	S	Z	W
	P	J	T	Q	A	W	R	U	A	X
	W	L	F	V	V	X	Y	P	I	L
	Q	B	G	X	O	Z	D	B	R	Y
	M	F	O	R	T	A	B	L	E	K
Блок 4	A	H	O	W	M	A	N	Y	T	I
	R	E	N	A	I	S	R	X	G	H
	I	D	E	N	J	O	Y	T	H	E
Блок 5	M	E	S	M	U	S	T	O	N	E
	S	D	Y	O	P	O	O	G	Y	T

Номер диска:	1	2	3	4	5	6	7	8	9	10
	V	Y	L	C	Q	R	L	K	T	O
	X	W	U	U	R	P	W	M	G	N
	Z	X	K	L	S	M	Q	N	H	U
	G	Z	B	T	W	U	X	Q	J	C
	H	M	R	Y	X	B	Z	R	K	D
	Y	A	C	B	M	D	C	V	M	F
	F	R	D	F	I	Y	A	W	P	G
	A	O	T	G	J	E	P	Y	Q	J
	R	S	F	S	Z	C	U	X	V	P
	I	U	G	H	A	V	H	Z	W	Q
	E	V	O	J	V	H	I	T	O	B
	K	I	N	D	O	F	S	A	U	N
Блок 6	G	O	I	N	H	O	W	L	O	N
	H	S	J	I	U	R	Q	C	U	V
	Y	U	A	N	P	P	X	D	X	W
	F	V	P	O	Q	M	Z	H	Z	X
	A	I	S	C	R	U	C	I	A	L
Блок 7	G	M	U	S	T	I	S	T	A	Y
	H	A	K	H	Y	J	E	Z	I	K
	Y	R	B	J	L	N	N	A	R	R
	F	O	R	D	E	G	R	E	E	S

Криптотекст может быть прочитан из нижних строк списков всех семи блоков. Теперь перепишем исходное сообщение и криптотекст, вставляя пробелы между словами и используя знаки препинания.

Исходное сообщение. What is the best temperature in sauna? How many times must one go in? How long must I stay?

Криптотекст. Hardly any rules. Feel comfortable, kid, enjoy. The kind of sauna is crucial for degrees.

Криптотекст не только удовлетворяет требованию Ришелье, но также отвечает на вопросы, поставленные в исходном сообщении! Ясно, что данное колесо Джефферсона специально было спроектировано для этой цели. Условия для такого проектирования исследуются в задаче 26.

□

Колесо Джефферсона реализует многоалфавитную подстановку. Рассмотрим версию, где мы заранее фиксируем сдвиг шифрования, т.е. фиксируется число i из 1, 2, ..., 25. Тогда криптотекст читается из i-х линий под исходным сообщением. В данном случае колесо может быть рассмотрено как многоалфавитная подстановка с периодом 10.

Рис. 1.8.

Ситуация несколько меняется, когда для каждого блока из 10 букв исходного сообщения сдвиг шифрования выбирается случайно, как это было сделано выше. Тогда после зашифрования каждых 10 букв исходного сообщения мы можем изменять подстановки для следующих 10 букв. Тем не менее существует всего 25 комбинаций подстановок, доступных для 10-элементных блоков.

Основная идея колеса Джефферсона — порождение многоалфавитных подстановок на основе независимо (более или менее) вращающихся

дисков, — является главной в изобретенных позже механических или электромеханических криптографических машинах. Удивительно, что большинство из этих машин вернулись к системе Цезаря со сдвигом один (по отношению к алфавитному порядку). Однако подстановки изменяются от буквы к букве, и, следовательно, можно рассматривать эту систему как систему Виженера, где длина ключевого слова огромна: в большинстве случаев 10^{10}. Таким образом, достижение успеха при криптоанализе с помощью метода Казизки маловероятно.

В качестве иллюстрации механических машин мы обсудим машину C–36 известного разработчика криптографических машин Бориса Хэйглина. Она известна, так же как M–209 Converter, и использовалась в армии США еще в начале пятидесятых годов.

Словесное описание механического устройства является крайне тяжелым, когда не доступен образец этого устройства. Маловероятно, что читатель имеет под рукой машину C–36, поэтому мы опишем функционирование данного устройства в абстрактной форме. Машина изображена на рис. 1.8. Ее основными компонентами являются шесть дисков, обычно называемых *роторами*, и цилиндр, называемый *клеткой*.

Рассмотрим 6×27-матрицу M, элементами которой являются 0 и 1. Потребуем также, чтобы в каждом из 27 столбцов матрицы M было не более двух единиц. Такие матрицы называются *кулачковыми матрицами*. Матрица

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

является примером кулачковой матрицы.

Очевидно, что если v является 6-разрядной строкой из нулей и единиц, то vM является 27-разрядной строкой с элементами из множества $\{0, 1, 2\}$. К примеру, если $v = (1, 0, 1, 1, 0, 0)$, то

$$vM = (0, 0, 1, 2, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 2) \ .$$

(Здесь мы использовали матрицу M, написанную выше.) Число элементов vM, отличных от нуля, называется *числом выталкиваний зубцов* v относительно M. В нашем примере оно равно 16. Обычно данное число является натуральным числом от 0 до 27.

Пошаговая матрица конструируется следующим образом. Построим 6 последовательностей чисел из множества $\{0,1\}$. Эти последовательности имеют соответствующие длины 17, 19, 21, 23, 25, 26 и начинаются с одной позиции. К примеру,

0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0									
0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0							
0	0	1	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	0	0	1	0	0	1	0	0	0	1	0	0	0	1			
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1

есть ступенчатая матрица. В отличие от кулачковой матрицы, для ступенчатой матрицы нет ограничений на позиции единиц.

Пошаговая матрица *генерирует* бесконечную последовательность 6-разрядных (строк) векторов следующим образом. Первые 17 векторов читаются прямо из столбцов. Таким образом,

$$(0,0,0,0,1,1) \qquad \text{и} \qquad (1,1,0,0,0,1)$$

являются первыми двумя векторами, порожденными с помощью написанной выше ступенчатой матрицы. Когда некоторая строка заканчивается, она вновь обращается к началу. Таким образом, векторами с 17-го по 47-й являются:

(0,0,0,0,0,0),	(0,0,0,0,0,0),	(1,0,0,1,0,1),	(1,0,0,0,0,0),
(0,1,0,0,0,0),	(0,1,0,0,0,0),	(0,1,0,1,0,0),	(1,1,1,0,0,0),
(0,1,0,0,0,0),	(0,0,0,0,1,1),	(0,0,0,0,0,1),	(0,0,0,0,1,1),
(0,0,0,0,0,0),	(0,0,1,0,0,0),	(0,0,0,0,0,0),	(1,0,0,0,0,0),
(1,0,0,0,0,0),	(0,0,0,0,0,0),	(0,0,0,1,0,0),	(1,0,0,0,0,0),
(1,0,0,0,0,0),	(0,0,0,1,0,0),	(0,0,0,0,0,0),	(0,1,0,0,0,0),
(1,1,0,0,0,1),	(0,1,0,1,0,0),	(0,1,0,0,0,0),	(0,1,0,0,0,0),
(0,0,1,0,0,1),	(0,0,0,1,0,0),	(0,0,0,0,0,0).	

Имея определенные кулачковую и ступенчатую матрицы, мы теперь можем сказать, как получается криптотекст. Для букв мы используем числовые коды: A получает номер 0, B получает номер 1 и т.д. Z получает номер 25. Как и прежде, арифметика ведется по модулю 26.

Обозначим через α i-ю букву исходного сообщения, а через h — число выталкиваний зубцов i-го вектора, порожденного ступенчатой матрицей, относительно кулачковой матрицы. Тогда α переводится в букву криптотекста

$$\gamma = h - \alpha - 1 \ .$$

Для примера рассмотрим исходное сообщение

GOVERNMENTOFTHEPEOPLEBYTHEPEOPLEANDFORTHEPEOPLE

для кулачковой и ступенчатой матриц, заданных выше. Числовое кодирование данного сообщения будет следующим. Мы используем здесь запятые только для ясности:

6, 14, 21, 4, 17, 13, 12, 4, 13, 19, 14, 5, 19, 7, 4, 15, 4, 14, 15, 11,
4, 1, 24, 19, 7, 4, 15, 4, 14, 15, 11, 4, 0, 13, 3, 5, 14, 17, 19, 7,
4, 15, 4, 14, 15, 11, 4.

Длина сообщения равна 47. Удалим пробелы между словами, как мы неоднократно это делали. При использовании криптографических машин пробелы иногда замещаются буквой Z.

Итак, мы вычисляем число выталкиваний зубцов для первых 47 векторов, порожденных ступенчатой матрицей. Это делается просто, так как первые 17 векторов можно видеть непосредственно из этой матрицы, а остальные уже найдены выше. Числа выталкиваний зубцов равны:

10, 17, 16, 9, 9, 9, 7, 0, 0, 0, 0, 12, 0, 0, 18, 7, 0, 0, 18, 7,
9, 9, 19, 14, 9, 10, 5, 10, 0, 0, 0, 7, 7, 0, 12, 7, 7, 12, 0, 9,
17, 19, 9, 9, 5, 12, 0.

По формуле $\gamma = h - \alpha - 1$ теперь вычислим числовые коды букв криптотекста:

3, 2, 20, 4, 17, 21, 20, 21, 12, 6, 11, 6, 6, 18, 13, 17, 21, 11, 2, 21,
4, 7, 20, 20, 1, 5, 15, 5, 11, 10, 14, 3, 6, 12, 8, 1, 18, 20, 6, 1,
12, 3, 4, 20, 15, 0, 21.

Поэтому мы получаем следующий криптотекст:

D C U E R V U V M G L G G S N R V L C V
E H U U B E P F L K O D G M I B S U G B
M D E U P A V.

Три появления PEOPLE в исходном тексте шифруются как RVLCVE, PELKOD и DEUPAV, тогда как 3 появления THE шифруются как GSN, UBF и GBM.

Сделаем несколько дополнительных замечаний, касающихся машины C–36. Роторы и клетки соответствуют ступенчатой и кулачковой матрицам. Любое переопределение ступенчатой матрицы осуществляется активизацией подходящих штифтов в роторах. Аналогично, любое

переопределение кулачковой матрицы получается позиционированием зубцов.

Кулачковая и ступенчатая матрицы образуют *ключ* для зашифрования с помощью С–36. Машина может быть рассмотрена как физическая реализация криптосистемы, описанной выше: она оперирует с переопределенным ключом после того, как активизируются подходящие штифты и позиционируются подходящие зубцы.

Уравнение $\gamma = h - \alpha - 1$ может быть записано также в виде $\alpha = h - \gamma - 1$. Это означает, что один и тот же ключ может использоваться для зашифрования и расшифрования и является причиной того, почему основное уравнение порождает систему типа Бьюфорта, а не типа Виженера–Цезаря.

Читатель может захотеть найти число всех возможных ключей для зашифрования с помощью С–36. При этом должно быть учтено дополнительное условие, налагаемое на кулачковую матрицу. Как будет показано ниже, не все возможные ключи являются хорошими с точки зрения обеспечения секретности.

Очевидно, что ступенчатая матрица генерирует векторы периодически. Следовательно, шифрование с помощью С–36 может быть рассмотрено как использование квадрата Бьюфорта с ключевым словом. Но какова длина ключевого слова? Обычно она намного длиннее любого допустимого сообщения. Следовательно, периодичность в криптотексте может и не появиться.

Действительно, длины строк в ступенчатой матрице попарно взаимно просты. Это означает, что только после

$$17 \cdot 19 \cdot 21 \cdot 23 \cdot 25 \cdot 26 = 101405850$$

шагов мы можем быть уверены, что опять вернемся в исходное состояние. В общем случае период не меньше данного числа, которое в действительности превышает число символов в достаточно объемной энциклопедии. Однако в конкретных случаях период может быть короче. К примеру, если ступенчатая матрица не содержит нулей, то генерируется только вектор (1,1,1,1,1,1), и, следовательно, период равен 1. Период будет коротким, если в кулачковой матрице имеется очень мало единиц или очень мало нулей в ступенчатой матрице. Поэтому такого выбора ключа нужно избегать.

Для того факта, что ступенчатая матрица состоит из шести строк, нет никаких математических оснований. Это число является компромиссом между секретностью и технической реализуемостью. Конечно, в целом период растет вместе с ростом числа строк. Число строк, очевидно, должно быть одинаковым в ступенчатой матрице и кулачковой

матрице. Большим преимуществом является также взаимная простота длин строк в ступенчатой матрице: это гарантирует максимальный период. Длины строк в ступенчатой и кулачковой матрицах произвольны. Кроме того, существует дополнительное требование, накладываемое на кулачковую матрицу. Физически это требование соответствует числу зубцов на шестерне в цилиндре.

Теперь должно быть очевидно, что метод Казизки или любой другой подобный подход неадекватен для криптоанализа C–36. Для изучения других криптоаналитических подходов читатель может обратиться к [BeP].

Некоторые известные криптографические машины, такие, как немецкая ENIGMA, американская SIGABA и японские RED и PURPLE времен второй мировой войны, являются электромеханическими. Основным блоком в них является диск в виде кодового колеса с проволочными перемычками внутри, называемый также *ротором*, по внешней и внутренней поверхностям которого равномерно распределены электрические контакты. Эти контакты позволяют объединять роторы. Как и для C–36, результирующая подстановка может меняться от буквы к букве.

Мы не хотим проводить более детальное обсуждение этих машин. Результирующие криптографические отображения являются существенно теми же, что и в случае C–36, по крайней мере с нашей точки зрения. Для более подробного изучения читатель отсылается к [BeP]. Что касается криптографических машин в целом, изобилие интересного материала содержит [Ka].

В оставшейся части этой главы рассмотрим наиболее широко используемую криптосистему всех времен: стандарт шифрования данных (DES — Data Encryption Standard) Национального бюро стандартов. Он был опубликован в 1977 — в [BeP] имеется перепечатка оригинальной публикации.

Описывающий стандарт DES алгоритм был разработан специально для электронных устройств для зашифрования и расшифрования данных. Идея “стандарта” в криптографии определенно является революционной. До опубликования DES не существовало публикаций, содержащих полный алгоритм для практического криптографического применения. Хотя мы делаем предположение, что криптоаналитик знает используемую криптосистему, большинство разработчиков криптосистем стараются скрыть детали их алгоритма. DES является замечательным исключением: алгоритм действительно опубликован. Это может быть рассмотрено как вызов всем тем, кто вскрывает системы!

Зашифрование и расшифрование, согласно DES, осуществляются

следующим образом. Сначала пользователи выбирают ключ, содержащий 56 случайных битов. Один и тот же ключ применяется в алгоритмах зашифрования и расшифрования и, конечно, хранится в секрете. Восемь битов, в позициях 8, 16,..., 64 добавляются в ключ таким образом, чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибки при обмене и хранении ключей. Таким образом, добавляемые биты определяются 56-ю исходными случайными битами, расположенными в позициях 1, 2,..., 7, 9,..., 15,..., 57,..., 63 ключа.

Эти 56 бит подвергаются следующей перестановке:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	64	53	45	37	29
21	13	5	28	20	12	4

Перестановка определяется двумя блоками C_0 и D_0 по 28 бит в каждом. Так, первые три бита C_0 (соответственно последние три бита D_0) есть 57, 49, 41 (соответственно 20, 12, 4) биты ключа.

Имея уже определенные блоки C_{n-1} и D_{n-1}, $n = 1, \ldots, 16$, мы строим блоки C_n и D_n одним или двумя *левыми сдвигами* из C_{n-1} и D_{n-1} согласно следующей таблице:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число левых сдвигов	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Один левый сдвиг означает смещение битов на один разряд влево: после одного левого сдвига биты в 28 разрядах являются битами, которые ранее находились в разрядах 2, 3,..., 28, 1. Таким образом, C_6 и D_6 получаются из C_5 и D_5 соответственно двумя левыми сдвигами.

Теперь мы готовы определить 16 перестановок K_n, $1 \leq n \leq 16$, битов ключа. Каждая K_n состоит из 48 бит, получаемых из битов $C_n D_n$

следующим образом:

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Итак, первыми (соответственно последними) тремя битами в K_n являются биты 14, 17, 11 (соответственно 36, 29, 32) в C_nD_n. Заметим, что 8 из 56 бит в C_nD_n отсутствуют в K_n.

Все предыдущие вычисления были в действительности предварительными: мы вычислили из ключа 16 последовательностей K_n по 48 бит в каждой. Теперь покажем, как зашифровать блок w, состоящий из 64 бит исходного сообщения. Сперва блок w подвергается *начальной перестановке*:

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Таким образом, после начальной перестановки мы имеем слово w', первыми тремя битами которого являются 58-й, 50-й и 42-й разряды слова w. Запишем $w' = L_0R_0$, где L_0 и R_0 оба содержат по 32 бит.

Имея построенные L_{n-1} и R_{n-1}, для $1 \le n \le 16$, мы определим L_n и R_n как

$$L_n = R_{n-1} \ ,$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n) \ ,$$

где $\oplus$ означает побитовое сложение по модулю 2, а f определяется ниже. Криптотекст c оригинального сообщения w теперь получается инверсией начальной перестановки к 64-битовому блоку $R_{16}L_{16}$.

Мы должны определить еще функцию f, но перед этим посмотрим, как происходит расшифрование. Оно осуществляется действительно очень просто: вышеуказанные уравнения могут быть переписаны в виде

$$R_{n-1} = L_n \ ,$$

$$L_{n-1} = R_n \oplus f(L_n, K_n) \ .$$

Мы можем, таким образом, "спуститься" от L_{16} и R_{16} к L_0 и R_0, после чего расшифрование очевидно!

Функция f получает из 32-битового блока R_{n-1} или L_n и 48-битового блока K_n (вспомните, как K_n был получен из ключа!) блок из 32 бит следующим образом. Первая переменная из 32 бит расширяется до 48 бит согласно следующей таблице:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Таким образом, первый бит из оригинального 32-битового блока появляется в позициях 2 и 48 нового 48-битового блока.

После такого расширения два 48-разрядных блока побитно складываются по модулю 2. Результирующий блок B из 48 бит делится на 8 6-битовых блоков: $B = B_1 B_2 \cdots B_8$. Каждый из этих восьми блоков B_i затем трансформируется в 4-битовый блок B_i' с помощью подходящей таблицы S_i, список которых приведен ниже:

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

Преобразование осуществляется следующим образом. Пусть B_7 равно 110010. Первый и последний разряды представляют число x, $0 \leq x \leq 3$. Аналогично средние 4 разряда представляют число y, $0 \leq y \leq 15$. В нашем случае $x = 2$ и $y = 9$. Строки и столбцы S_7 нумеруются числами x и y. Таким образом, пара (x, y) однозначно определяет число. В нашем случае этим числом является 15. Тогда для его двоичного представления мы получим $B'_7 = 1111$.

Значение f теперь получается применением перестановки

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

к результирующему 32-битному блоку $B'_1B'_2\cdots B'_8$. Это завершает определение функции f, так же как и наше описание алгоритмов зашифрования и расшифрования, соответствующих DES.

DES-алгоритмы работают очень быстро на подходящем оборудовании. С другой стороны, криптоанализ приводит к многочисленным нелинейным системам уравнений. Возникающие при этом задачи будут по крайней мере NP-полными (см. приложение А). Тем не менее имеется предположение, что сконструированная для этих целей машина может перебрать все возможные ключи. Специальная аппаратура будет осуществлять поиск среди всех 2^{56} ключей со скоростью 10^{12} ключей в секунду: 10^6 чипов, каждый из которых ищет в разных частях ключевого пространства со скоростью один ключ в микросекунду. Оценки стоимости такого оборудования могут сильно отличаться. Более подробно эти проблемы рассмотрены, например, в [De].

До сих пор было установлено несколько свойств DES-отображений. Интересное свойство, касающееся симметрии, рассмотрено в задаче 16.

Также DES обладает очень желательной с точки зрения секретности особенностью: незначительное изменение исходного сообщения или ключа приводит к большим изменениям в криптотексте. Детализирующие рисунки, касающиеся этого *лавинообразного эффекта*, могут быть найдены в [Kon].

Рис. 1.9.

Глава 2

Идея открытых ключей

2.1. Некоторые улицы являются односторонними

Подумаем немного о криптосистемах, представленных в гл. 1, а также о любых других подобных системах. Для криптоаналитика, знающего алгоритм зашифрования, не возникает никаких проблем в процессе расшифрования. Ключи зашифрования и расшифрования совпадают даже в такой сложной системе, как DES. Поэтому, работая с одной из представленных систем и открывая алгоритм зашифрования, вы тем самым рассекречиваете свои сообщения.

Но это происходит не всегда. Существуют системы, для которых можно раскрыть алгоритм зашифрования без уменьшения секретности в целом. Это означает, что криптоаналитик также будет знать данный алгоритм. Но тем не менее он еще не в состоянии расшифровать ваш криптотекст. Все это сказано о *криптографии с открытым ключом*: алгоритм зашифрования может быть опубликован.

Идея открытых ключей была представлена Диффи и Хеллманом [DH]. Идея очень проста по своей сути, хотя и революционна. Почему же такая простая идея была представлена так поздно — в середине семидесятых — за очень длинную историю криптографии? Как обеспечивается безопасность при раскрытии алгоритма зашифрования? Как можно реализовать эту прекрасную идею?

Ответ на первый вопрос будет легким: теория сложности была развита только недавно. Эта теория дает нам информацию о сложности различных вычислений, скажем, о том, как много времени будет затрачено для вычислений на лучших компьютерах. Такая информация

является очень важной в криптографии.

Это приводит нас ко второму вопросу. Конечно, алгоритм зашифрования открывает в математическом смысле и алгоритм расшифрования, потому что они “обратны” друг другу. Предположим тем не менее, что потребуются сотни лет работы криптоаналитика для раскрытия алгоритма расшифрования из алгоритма зашифрования. Тогда раскрытие алгоритма зашифрования ничему не угрожает. Таким образом понимается “безопасность” во втором вопросе.

Что касается вопроса о реализации идеи открытых ключей, далее он будет рассмотрен более подробно. Сделаем здесь лишь несколько предварительных замечаний.

В математике, так же как и в реальной жизни, существуют односторонние улицы. Легко добраться по такой улице из A в B, тогда как практически невозможно добраться из B в A. Шифрование рассматривается как направление от A к B. Хотя мы можем двигаться в этом направлении, мы не в состоянии двигаться в обратном направлении — расшифрования.

Возьмем телефонный справочник большого города. Легко найти номер любого абонента. Но, с другой стороны, тяжело — можно сказать, невозможно! — найти абонента, имеющего заданный номер. Справочник состоит из нескольких толстых томов. В принципе вы должны аккуратно пролистать каждый из них.

Это дает идею для криптосистем с открытым ключом. Шифрование является контекстносвободным: буква переходит в букву. Для каждой буквы исходного сообщения случайно из справочника выбирается имя, начинающееся с этой буквы. Соответствующий телефонный номер образует шифр для данного частного случая появления буквы. Таким образом, система является многоалфавитной: два различных появления одной буквы с очень малой вероятностью шифруются одинаково. Шифрование сообщения COMETOSAUNA может быть таким:

Сообщение	Выбранное имя	Криптотекст
C	Cobham	7184142
O	Ogden	3529517
M	Maurer	9372712
E	Engeler	2645611
T	Takahashi	2139181
O	Orwell	5314217

Сообщение	Выбранное имя	Криптотекст
S	Scott	3541920
A	Adleman	4002132
U	Ullman	7384502
N	Nivat	5768115
A	Aho	7721443

Таким образом, весь криптотекст получается написанием один за другим всех номеров, появляющихся в правом столбце. При этом, конечно, номера записываются в том порядке, как они указаны в списке.

Заметим, что алгоритм зашифрования не является детерминированным. Бесконечно много криптотекстов можно получить из одного и того же исходного сообщения. С другой стороны, каждый криптотекст порождает только один исходный текст.

Легальный получатель исходного сообщения может иметь справочник, составленный согласно возрастанию номеров. Такой справочник делает процесс расшифрования очень легким. Согласно терминологии, обсуждаемой более подробно далее, обратный справочник является *лазейкой*, известной только легальным пользователям системы.

Не зная лазейки, т. е. не имея на руках копии обратного справочника, криптоаналитик затратит на расшифрование очень много времени. Это так, несмотря на тот факт, что алгоритм зашифрования опубликован и криптоаналитик, в принципе, знает, как ему следует интерпретировать перехваченные числовые последовательности.

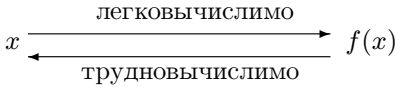


Рис. 2.1.

Поиск по всему справочнику, вероятно, будет очень долог. Конечно, криптоаналитик может также попытаться дозвониться по всем номерам из криптотекста и спросить имена. Успех этого метода сомнителен — можно получить сердитый ответ или в очень многих случаях не получить ответа совсем. Метод становится заведомо непригодным при использовании старого справочника.

Система, основанная на телефонном справочнике, годится только в качестве начальной иллюстрации, а не для серьезного применения. Действительно, не так трудно составить “обратные” справочники.

Идея криптографии с открытым ключом тесно связана с идеей *односторонних функций*. По заданному аргументу x легко вычислить значение функции $f(x)$, тогда как определение x из $f(x)$ трудновычислимо. Здесь "трудновычислимость" понимается в смысле теории сложности (см. приложение А). Ситуация изображена на рис. 2.1.

Мы говорим о $f(x)$ как о функции. Однако рис. 2.1 понимается в более широком смысле: допускаются также недетерминированные алгоритмы зашифрования, такие, как для примера с телефонным справочником.

Определение x из $f(x)$ трудновычислимо только для криптоаналитика. Легальный получатель имеет подходящую лазейку. Далее такие односторонние функции будем называть *криптографическими*.

Упомянем по этому поводу, что ни одного примера криптографической односторонней функции не известно. Зато существует много криптографических функций $f(x)$, таких, что:

1. Легко вычислить $f(x)$ из x.

2. Определение x из $f(x)$, вероятно, будет трудновычислимым.

Однако не имеется доказательств для трудновычислимости, требуемой в пункте 2. Это отражает тот факт, что очень тяжело получить высокие нижние оценки в теории сложности. Очень непросто показать, что трудновычислимость алгоритма, используемого нами, является неслучайной.

С точки зрения криптографии с открытым ключом вполне подходят функции, удовлетворяющие требованиям 1 и 2. В типичных криптосистемах с открытым ключом только прямой криптоанализ основан на вычислении x из $f(x)$. Могут существовать и другие, более гениальные криптоаналитические методы, избегающие этого вычисления. Таким образом, криптоаналитик может достичь успеха даже в том случае, когда доказано, что нахождение x из $f(x)$ трудновычислимо.

Эти выводы будут обсуждаться в следующем примере.

Пример 2.1. Уточним определение односторонних функций. Задача называется *трудновычислимой*, если нет алгоритма для решения данной задачи с полиномиальным временем работы. Если же такой алгоритм существует, то задача называется *вычислимой*. *Легковычислимыми* будут называться задачи, имеющие алгоритмы со временем работы, представимым в виде полинома низкой степени относительно входного размера задачи, а еще лучше алгоритмы с линейным временем работы. *NP-полные* задачи рассматриваются как трудновычислимые.

Все эти определения соответствуют стандартной терминологии из теории сложности. Для более детального рассмотрения читатель отсылается к приложению А. Заметим, что традиционная теория сложности не идеальна с точки зрения криптографии. Сложность всей задачи в традиционной теории сложности определяется сложностью "в самом худшем случае". Так как такие "худшие случаи" могут появляться крайне редко, то для криптографии информация о средней сложности является более существенной.

Функция $f(x)$ будет односторонней, если перевод x в $f(x)$ легок, а обратный перевод из $f(x)$ в x трудновычислим. Второе требование часто заменяется более слабым условием: обратный перевод, вероятно, будет трудновычислимым (это является вышеуказанным условием 2).

Наш пример основан на *задаче о рюкзаке*. Задан набор

$$(a_1, a_2, \ldots, a_n) = A$$

n различных положительных целых чисел и еще одно положительное целое число k. Задачей является нахождение таких a_i, если это возможно, сумма которых равна k.

Рис. 2.2.

В простейшем случае k указывает размер рюкзака, а каждое из чисел a_i указывает размер предмета, который может быть упакован в рюкзак. Задачей является нахождение такого набора предметов, чтобы рюкзак был полноситью заполнен.

В качестве иллюстрации рассмотрим число 3231 и набор из 10 чисел

$$(43,\ 129,\ 215,\ 473,\ 903,\ 302,\ 561,\ 1165,\ 697,\ 1523)\ .$$

Заметим, что

$$3231 = 129 + 473 + 903 + 561 + 1165\ .$$

Таким образом, мы нашли решение. Ситуация изображена на рис.2.2.

В принципе решение всегда может быть найдено полным перебором подмножеств A и проверкой, какая из их сумм равна k. В нашем случае это означает перебор $2^{10} = 1024$ подмножеств (включая при этом и пустое множество). Это вполне осуществимо.

Но что будет, если существует несколько сотен чисел a_i? В нашем примере $n = 10$, чтобы не усложнять изложение. В реальных условиях пример будет иметь, скажем, 300 a_i-х. Суть здесь в том, что неизвестны алгоритмы, имеющие существенно меньшую сложность по сравнению с полным перебором. Поиск среди 2^{300} подмножеств не поддается обработке. В самом деле, задача о рюкзаке известна как NP-полная.

Наш n-набор A определяет функцию $f(x)$ следующим образом. Любое число x в интервале $0 \le x \le 2^n - 1$ может быть задано двоичным представлением из n разрядов, где при необходимости добавляются начальные нули. Таким образом, 1, 2 и 3 представляются в виде $0\ldots01, 0\ldots010, 0\ldots011$, тогда как $1\ldots111$ есть представление для $2^n - 1$. Теперь определим $f(x)$ как число, получаемое из A суммированием всех таких a_i, что соответствующий разряд в двоичном представлении x равен 1. Так,

$$\begin{array}{lcll} f(1) & = & f(0\ldots001) & = & a_n, \\ f(2) & = & f(0\ldots010) & = & a_{n-1}, \\ f(3) & = & f(0\ldots011) & = & a_{n-1} + a_n, \end{array}$$

и т. д. Используя векторное умножение, мы можем записать

$$f(x) = AB_x,$$

где B_x — вектор-столбец двоичного представления x.

Наше предыдущее равенство (см. также рис. 2.2) может быть записано в виде

$$f(364) = f(0101101100) = 129 + 473 + 903 + 561 + 1165 = 3231\ .$$

Дальнейшие значения функции определяются наборами из 10 разрядов

$$
\begin{aligned}
f(609) &= f(1001100001) = 43 + 473 + 903 + 1523 = 2942,\\
f(686) &= f(1010101110) = 43 + 215 + 903 + 561 + 1165 + 697 = 3584,\\
f(32) &= f(0000100000) = 903,\\
f(46) &= f(0000101110) = 903 + 561 + 1165 + 697 = 3326,\\
f(128) &= f(0010000000) = 215,\\
f(261) &= f(0100000101) = 129 + 1165 + 1523 = 2817,\\
f(44) &= f(0000101100) = 903 + 561 + 1165 = 2629,\\
f(648) &= f(1010001000) = 43 + 215 + 561 = 819.
\end{aligned}
$$

Эти конкретные значения потребуются ниже.

Функция $f(x)$ определялась n-набором A. Очевидно, что если мы в состоянии вычислить x из $f(x)$, то практически за то же время будет решена задача о рюкзаке: по x немедленно вычисляется его двоичное представление, которое в свою очередь дает компоненты набора A, входящие в сумму для $f(x)$. С другой стороны, вычисление $f(x)$ из x является легким. Так как задача о рюкзаке NP-полна, $f(x)$ является хорошим кандидатом для односторонней функции. Конечно, надо потребовать, чтобы n было достаточно большим, скажем, не менее 200. Ниже будет показано, что функция $f(x)$ является также криптографической.

Сначала рассмотрим, как "рюкзачные векторы" A могут быть использованы в качестве основы криптосистемы. Исходное сообщение вначале кодируется и разбивается на n-разрядные блоки. Если это необходимо, последний блок дополняется в конце нулями. Каждый из n-разрядных блоков теперь шифруется с помощью вычисления значения функции f для этого блока.

Если исходное сообщение написано по-английски, естественным способом кодирования является замена каждой буквы двоичным представлением ее порядкового номера в алфавите. Для этих целей потребуется пять битов. В следующей таблице нумерация букв начинается с 1, тогда как пробел между двумя словами задается числом 0.

Буква	Число	Двоичное представление
Пробел	0	00000
А	1	00001
В	2	00010
С	3	00011
Д	4	00100

Буква	Число	Двоичное представление
E	5	00101
F	6	00110
G	7	00111
H	8	01000
I	9	01001
J	10	01010
K	11	01011
L	12	01100
M	13	01101
N	14	01110
O	15	01111
P	16	10000
Q	17	10001
R	18	10010
S	19	10011
T	20	10100
U	21	10101
V	22	10110
W	23	10111
X	24	11000
Y	25	11001
Z	26	11010

Рассмотрим предыдущий 10-набор и исходное сообщение SAUNA AND HEALTH. Так как шифруемые блоки состоят из 10 разрядов, то деление на блоки нашего исходного текста даст:

SA UN Aпробел AN Dпробел HE AL TH

Соответствующие 8 двоичных последовательностей:

1001100001 ,
1010101110 ,
0000100000 ,
0000101110 ,
0010000000 ,
0100000101 ,
0000101100 ,
1010001000 .

Они в точности являются аргументами значений рассмотренной выше функции f, следовательно, криптотекстом будет 8-набор

(2942, 3584, 903, 3326, 215, 2817, 2629, 819) .

Определенная таким образом на основе функции рюкзака $f(x)$ криптосистема еще не является системой с открытым ключом. Действительно, мы можем использовать ее как классическую систему. Тогда криптоаналитик находит n-набор A и после этого решает еще и задачу о рюкзаке.

Если криптоаналитик может использовать в качестве условия анализа "избранный исходный текст", то криптоаналитик использует исходные сообщения с ровно одним появлением единицы и легко находит вектор A.

Но для расшифрования легальный получатель также должен решать задачу о рюкзаке. Это означает, что расшифрование одинаково трудно (и требует решения NP-полной задачи) и для криптоаналитика, и для легального получателя. Такая ситуация нежелательна: она показывает, что криптосистема очень несовершенна. В хороших криптосистемах расшифрование значительно труднее для криптоаналитика, чем для легального получателя.

Прежде чем улучшить данную криптосистему и преобразовать ее в систему с открытым ключом, сделаем один вывод. Никогда нельзя получить два различных исходных сообщения из одного и того же криптотекста. Это означает, что не существует двух равных сумм, по-разному сформированных из элементов набора A. Суммы могут иметь одинаковое или разное число слагаемых, но каждый элемент из A может быть использован только один раз. Можно показать, что 10-набор, обсуждаемый выше, обладает этим свойством. А 5-набор (17, 103, 50, 81, 33) не обладает этим свойством. Согласно этому набору, криптотекст

$$(131,\ 33,\ 100,\ 234,\ 33)$$

может быть расшифрован как SAUNA и FAUNA — здесь есть высокая степень неопределенности! Дальнейшие попытки расшифрования этого криптотекста будут результативными, если мы имеем символ исходного сообщения, закодированный двоичной последовательностью 11011.

Теперь преобразуем криптосистему, основанную на n-наборе A, в систему с открытым ключом. Сначала сделаем несколько замечаний, а затем вернемся к нашей числовой иллюстрации.

Существуют подклассы легких задач укладки рюкзака. Один из них получается из сверхрастущих n-наборов A. n-Набор

$$A = (a_1, a_2 \ldots a_n)$$

называется *сверхрастущим*, если каждое следующее число в нем

больше суммы всех предыдущих, т.е.

$$a_j > \sum_{i=1}^{j-1} a_i \qquad \text{для } j = 2, \ldots, n \ .$$

Нет необходимости в полном переборе при решении соответствующей задачи о рюкзаке — достаточно просмотреть A один раз справо налево. Для заданного k (размера рюкзака) мы сначала проверяем $k \geq a$. Если ответом будет "нет", a_n не может входить в искомую сумму. Если же ответом будет "да", то a_n *должно* входить в сумму. Это следует из того факта, что все оставшиеся a_i-е в сумме дают число, меньшее k. Определим

$$k_1 = \begin{cases} k, & \text{если } k < a_n, \\ k = k - a_n, & \text{если } k \geq a \end{cases}$$

и повторим ту же самую процедуру для k_1 и a_{n-1}. Так можно дойти до a_1. Алгоритм также показывает, что для любого k задача рюкзака имеет не более одного решения, при условии, что A является сверхрастущим.

Если мы раскрываем сверхрастущий набор A как основу криптосистемы, то расшифрование будет одинаково легким для криптоаналитика и легального получателя. Чтобы избежать этого, мы "взболтаем" A таким образом, чтобы результирующий набор B не являлся сверхрастущим и выглядел как произвольный вектор рюкзака. В действительности он только выглядит как произвольный, потому что очень немногие векторы рюкзака могут быть получены таким способом: используемое нами взбалтывание является модульным умножением. В самом деле, мы использовали модульную арифметику уже много раз в гл. 1. Читатель, не знакомый с понятием сравнения, может обратиться к приложению В.

Выберем целое $m > \sum a_i$. Так как A сверхрастущий, то m велико по сравнению со всеми числами из A. Выберем другое целое t, не имеющее общих множетелей с m. m и t являются *модулем* и *множителем*. Такой выбор t гарантирует, что найдется другое целое t^{-1}, такое, что $tt^{-1} \equiv 1 \pmod{m}$. Целое t^{-1} можно назвать *обратным* к t. Обратный элемент может быть легко вычислен по t и m.

Теперь образуем произведения ta_i, $i = 1, \ldots, n$, и сведем их по модулю m: пусть b_i — наименьший положительный остаток ta_i по модулю m. Результирующий вектор

$$B = (b_1, b_2 \ldots, b_n)$$

открывается как ключ зашифрования. Алгоритм зашифрования для n-разрядных блоков исходного сообщения описан выше.

Числа t, t^{-1} и m хранятся как *секретная лазейка*. Перед сравнением ситуации с точки зрения криптоаналитика и легального получателя вернемся к уже рассмотренному примеру.

Легко видеть, что наш предыдущий 10-набор (обозначенный теперь через B)

$$B = (43,\ 129,\ 215,\ 473,\ 903,\ 302,\ 561,\ 1165,\ 697,\ 1523)$$

получается с помощью модульного умножения с $m = 1590$ и $t = 43$ из сверхрастущего вектора рюкзака

$$A = (1,\ 3,\ 5,\ 11,\ 21,\ 44,\ 87,\ 175,\ 349,\ 701)\ .$$

Проверим это более подробно.

Первые пять чисел в B получаются из соответствующих чисел в A простым умножением на 43 — сведение по модулю не нужно. (В реальных ситуациях даже первые числа не должны быть очень маленькими, иначе легко обнаружить множитель.) Следующие вычисления позволяют получить оставшиеся пять чисел в B.

$$\begin{aligned}
43 \cdot 44 &= 1892 = 1590 + 302\ , \\
43 \cdot 87 &= 3741 = 2 \cdot 1590 + 561\ , \\
43 \cdot 175 &= 7525 = 4 \cdot 1590 + 1165\ , \\
43 \cdot 349 &= 15007 = 9 \cdot 1590 + 697\ , \\
43 \cdot 701 &= 30143 = 18 \cdot 1590 + 1523\ .
\end{aligned}$$

Заметим далее, что t и m взаимно просты. Действительно,

$$43 \cdot 37 = 1591 \equiv 1 \pmod{1590}\ .$$

Следовательно, $t^{-1} = 37$.

Теперь отыщем легкий алгоритм расшифрования для легального получателя. Рассмотрим сначала основной случай, где A является сверхрастущим вектором, а B получается из A умножением каждого числа в A на $t \pmod{m}$. Так как легальный получатель знает t^{-1} и m, он в состоянии найти A из открытого ключа B. После получения блока c' криптотекста, который является целым числом, легальный получатель вычисляет $t^{-1}c'$ и его наименьший положительный остаток $c \pmod{m}$. При расшифровании он решает легкую задачу укладки рюкзака, определяемую A и c. Решением является единственная последовательность p из n битов. Оно также является и правильным блоком исходного сообщения, так как любое решение p' задачи укладки рюкзака, определяемой B и c', должно равняться p. Действительно,

$$c \equiv t^{-1}c' = t^{-1}Bp' \equiv t^{-1}Ap' \equiv Ap' \pmod{m}\ .$$

Отметим, что $Ap' < m$, так как $m > a_1 + a_2 + \ldots + a_n$. Это означает, что указанное выше сравнение может быть переписано в виде уравнения $c = Ap'$. Так как задача укладки рюкзака, определяемая A и c, не может иметь несколько решений, то $p' = p$.

Итак, каким образом легальный получатель может вручную расшифровать криптотекст

$$(2942,\ 3584,\ 903,\ 3326,\ 215,\ 2817,\ 2629,\ 819)\ ,$$

полученный ранее? Умножая на $t^{-1} = 37$, он/она получает первое число

$$37 \cdot 2942 = 108854 = 68 \cdot 1590 + 734 \equiv 734 \pmod{1590}\ .$$

Произведя аналогичные вычисления, он получает восьмерку

$$(734,\ 638,\ 21,\ 632,\ 5,\ 879,\ 283,\ 93)\ .$$

Число 734 и сверхрастущий набор A дают 10-разрядную последовательность 1001100001. Действительно, так как $734 > 701$, то последний бит должен быть равен 1. Числа из A теперь сравниваются с разностью $734 - 701 = 33$. Первое число, справа налево, меньшее чем 33, есть 21. Следующее число 11 меньше разности $33 - 21 = 12$. Наконец, первое число 1 равно разности $12 - 11$. Позиции 1, 11, 21 и 701 в A есть соответственно 1, 4, 5 и 10.

Таким же образом числа $638, \ldots, 93$ дают другие семь 10-разрядных вышеупомянутых последовательностей. Декодируя все восемь последовательностей, легальный получатель вычисляет исходное сообщение SAUNA AND HEALTH.

□

Приведенный выше пример 2.1 составляет главную часть этого параграфа. Основные принципы для построения криптосистем с открытым ключом будут подробно изложены в следующем параграфе. Криптосистема, основанная на сверхрастущем векторе рюкзака, служит примером и дает детальную иллюстрацию этих принципов. С другой стороны, такая криптосистема не очень надежна: в гл. 3 будет обсуждаться алгоритм, вскрывающий данную систему за полиномиальное время. Алгоритм основан на том факте, что не обязательно для криптоаналитика нахождение истинных значений множителя t и модуля m, которые действительно используются при разработке системы. Достаточно найти любые t' и m', такие, что умножение открытого вектора на $t'^{-1} \pmod{m}$ дает сверхрастущий вектор. Таким образом, криптоаналитик может действительно вскрыть систему с помощью *предварительного криптоанализа*, после того как был раскрыт ключ зашифрования. Так как открытые ключи зашифрования используются в течение

некоторого времени, существует возможность для предварительного криптоанализа, в то время как криптоаналитик действует в спешке после перехвата важных зашифрованных сообщений.

"Односторонние улицы" — это то, чем занимается криптография с открытым ключом. Читатель может придумать примеры "односторонних улиц" из различных сфер жизни. Вот один из очень типичных примеров. Устройство, изображенное на рис. 2.3, является ловушкой для рыбной ловли в северных странах.

Рис. 2.3.

Рыбе очень легко забраться в клетку. Форма входа ведет рыбу внутрь — здесь в качестве приманки может находиться небольшая рыбка. С другой стороны, рыбе очень трудно найти путь назад, хотя в принципе бегство возможно. Легальный пользователь, т.е. рыбак, берет рыбу через отверстие в верхней части клетки.

2.2. Как реализовать идею

Этот параграф содержит некоторые основные принципы построения криптосистем с открытым ключом. Зная ключ зашифрования E_k, нельзя вычислить ключ расшифрования D_k, т. е. нахождение D_k из E_k трудновычислимо, по крайней мере для почти всех ключей k.

Следующий механический аналог изображает разницу между классическими криптосистемами и криптосистемами с открытым ключом. Пусть информация пересылается в ящике с закрытыми замками. Тогда зашифрование, согласно классической криптосистеме, соответствует запиранию ящика на висячий замок и посылке ключа по некоторому

абсолютно секретному каналу, к примеру с помощью агента класса Джеймса Бонда. *Управление ключом* всегда является основной задачей и часто очень трудной при использовании классических криптосистем.

Криптография с открытым ключом соответствует наличию открытых висячих замков, снабженных вашим именем, доступных в таких местах, как почта. Тот, кто хочет послать вам сообщение, закрывает ящик с вашим замком и посылает его вам. Только вы имеете ключ для открытия замка.

Рис. 2.4.

Для классических систем подходит следующая модификация основной процедуры открытых ключей. Обозначим через $E_A, E_B, \ldots$ процедуры зашифрования пользователей $A, B, \ldots$. Аналогично обозначим процедуры расшифрования через $D_A, D_B, \ldots$. Потребуем далее, чтобы криптосистема являлась *коммутативной*: порядок в любой композиции $E_A, E_B, D_A, D_B, \ldots$ не важен. Если A хочет послать сообщение w к B, то используется следующий протокол:
(1) A посылает $E_A(w)$ к B .
(2) B пересылает $E_B(E_A(w))$ к A .
(3) A посылает $D_A(E_B(E_A(w))) = D_A(E_A(E_B(w))) = E_B(w)$ к B .

(4) B расшифровывает $D_B(E_B(w)) = w$.

Возвращаясь к нашему примеру с висячими замками, отметим, что для данного протокола открытые висячие замки не нужно распределять заранее. Сначала A посылает к B ящик, закрытый висячим замком A. Затем B посылает обратно к A ящик, теперь закрытый еще и замком B. A открывает висячий замок E_A и посылает ящик обратно к B. Теперь B может открыть его. Таким образом, ящик всегда закрыт, по крайней мере, на один замок. В этом примере нет проблем в управлении ключами: ключи не распределяются совсем. (См. рис. 2.4.)

Протокол, описанный выше, обеспечивает безопасность против *пассивного* перехватчика. Однако *активный* перехватчик C может замаскироваться под B. Тогда A не может узнать, кто в действительности находится на другой стороне. Под *пассивным перехватчиком* мы понимаем криптоаналитика, который пытается только получать всю возможную информацию для расшифрования важных сообщений. *Активный перехватчик* маскируется под предполагаемого получателя сообщения и соответственно пересылает информацию оригинальному отправителю.

Теперь мы готовы перечислить *основные принципы построения криптосистем с открытым ключом*.

Шаг 1: Начинаем с трудной задачи P. P должна быть труднорешаемой в смысле теории сложности: не должно существовать алгоритма, который решает все варианты задачи P за полиномиальное время относительно размера задачи. Предпочтительнее, чтобы не только сложность “в худшем случае”, но и сложность в среднем задачи P была достаточно высокой.

Шаг 2: Выделяется легкая подзадача P_{easy} из P. P_{easy} должна решаться за полиномиальное время, предпочтительнее даже за линейное время.

Шаг 3: “Перетасовать или взболтать” P_{easy} таким образом, чтобы результирующая задача $P_{shuffle}$ не имела никакого сходства с P_{easy}. Задача $P_{shuffle}$ должна, по крайней мере, выглядеть как оригинальная труднорешаемая задача P.

Шаг 4: Открывается $P_{shuffle}$ с описанием, как она может быть использована в качестве ключа зашифрования. Информация о том, как из $P_{shuffle}$ получить P_{easy}, держится в секрете как *секретная лазейка*.

Шаг 5: Детали криптосистемы конструируются таким способом, чтобы алгоритм расшифрования был существенно различным для криптоаналитика и легального получателя. Пока первый решает $P_{shuffle}$ (имеющую вид труднорешаемой задачи P), последний может использовать секретную лазейку и решать только P_{easy}.

Наше описание шагов 1–5, конечно, находится еще на очень абстрактном уровне. Качество результирующей криптосистемы с открытым ключом зависит от используемых в ней деталей. Существует много вопросов, на которые надо ответить. Как использовать $P_{shuffle}$ для зашифрования? Насколько легкой является P_{easy}? Что составляет секретную лазейку? В частности, возможно ли найти секретную лазейку с помощью предварительного криптоанализа? Может ли вариант $P_{shuffle}$ быть легким только случайно? И так далее. Ниже мы вернемся к поставленным вопросам.

Теперь вспомним пример 2.1 из предыдущего параграфа. Он является очень типичной иллюстрацией шагов 1–5. Задача укладки рюкзака является NP-полной, поэтому это очень подходящий выбор труднорешаемой задачи, с помощью которой может быть построена система. Задача укладки рюкзака со сверхрастущим набором является легкой подзадачей P. Модульное умножение дает подходящий способ для "взбалтывания". В гл. 3 мы еще вернемся к этой задаче и обсудим, насколько в действительности она подходит для построения хорошей криптосистемы с открытым ключом. Это обсуждение будет связано как с возможностями криптоаналитика, так и с некоторыми модифицированными криптосистемами. В общем рюкзачные векторы дают естественный и удобный для применения алгоритм зашифрования.

Шаги 1–5 криптографии с открытым ключом обладают важным свойством — универсальностью: субъект предмета или область применения никак не оговаривается. В принципе задачи могут быть почти обо всем. Примеры будут показаны в последующих главах. Тем не менее многие задачи, наиболее подходящие в качестве основы для криптосистем с открытым ключом, имеют связь с теорией чисел. Мы уже рассмотрели один пример из теории чисел: задачу укладки рюкзака. Наиболее известная криптосистема с открытым ключом — RSA — также основана на теории чисел. Произведение двух огромных простых чисел может быть открыто без указания самих чисел. Односторонняя функция или секретная лазейка может быть сформулирована в этих терминах. Более подробно эта система будет рассмотрена в гл. 4.

Очень часто о задачах, лежащих в основе систем с открытым ключом, известно очень мало или вообще ничего, что вполне присуще природе криптографии с открытым ключом. Так, RSA является очень успешной, хотя сложность факторизации точно не определена. С другой стороны, некоторые криптосистемы с открытым ключом, основанные на стандартных труднорешаемых задачах (NP-полных, например), обманули ожидания.

Для дальнейших ссылок мы дадим список некоторых фундаменталь-

ных задач теории чисел, для которых до сих пор не удалось определить их сложность. Действительно, ни для одной из перечисленных ниже задач неизвестно полиномиальных алгоритмов и они не отнесены ни к какому естественному классу сложности. Представленные задачи очень активно используются в криптографии с открытым ключом. Также известны некоторые способы сведения задач друг к другу: какие из них "легче" и какие "труднее".

FACTOR(n). Найти разложение на множители n.

PRIMALITY(n). Решить, является ли n простым числом.

FIND-PRIME($> n$). Найти простое число $> n$.

SQUAREFREENESS(n). Решить, делит или нет квадрат простого числа число n.

QUAD-RESIDUE(a, n). Решить, выполняется или нет сравнение $x^2 \equiv a \pmod{n}$ для некоторого x.

SQUAREROOT(a, n). Найти, если возможно, такое число x, что $x^2 \equiv a \pmod{n}$.

DISCRETE-LOG(a, b, n). Найти, если возможно, такое x, что $a^x \equiv b \pmod{n}$.

Читатель, разбирающийся в теории чисел, может подумать о некоторых естественных спообах сведения представленных задач. К примеру, если мы в состоянии факторизовать n, мы также сможем ответить, является ли n простым. В действительности задача проверки чисел на простоту существенно проще факторизации, потому что существует много легкопроверяемых критериев следующего вида: если n является простым, то выполняется условие A (к примеру, сравнение). Следовательно, если A не выполняется, то мы можем заключить, что n составное, не разлагая n на множители.

С теоретической точки зрения желательно было бы установить некоторые нижние оценки времени работы криптоаналитика для вскрытия системы с открытым ключом. К несчастью, такие нижние оценки не известны для большинства широко используемых систем с открытым ключом. К примеру, FACTOR(n) может иметь низкий полиномиальный размер, что приведет к краху RSA и подобных систем. С другой стороны, маловероятно, что FACTOR(n) имеет низкий полиномиальный размер. К тому же люди исследуют FACTOR(n) (более или менее интенсивно) уже более двух тысяч лет.

Теперь обсудим некоторые положения теории сложности, которые проливают свет на эти проблемы: не существует доказанных нижних

оценок времени работы криптоаналитика для анализа системы с открытым ключом. На самом деле, наше Золотое правило может быть распространено и на криптосистемы с открытым ключом.

Золотое правило для создателей криптосистем с открытым ключом. Проверьте вашу систему на практике с разных точек зрения. Не надейтесь доказать замечательные результаты, касающиеся безопасности вашей системы.

Читатель, не знакомый с основами теории сложности, может обратиться к приложению А. Обычно предполагается, что $P \neq NP$. Это означает, что NP-полные задачи являются труднорешаемыми. Следовательно, если мы можем показать, что криптоанализ системы с открытым ключом является NP-полной задачей, мы установим его трудновычислимость. Однако следующий довод показывает, что этот случай является маловероятным.

Ключ зашифрования публикуется. Объединим этот факт с требованием, предъявляемым к любой системе, классической и с открытым ключом: зашифрование по ключу и известному исходному тексту не представляет трудности. (В противном случае криптосистема будет очень громоздкой для использования!) Следовательно, в любой подходящей системе с открытым ключом *задача криптоанализа лежит в классе* NP. Имея криптотекст, аналитик вначале догадывается об исходном сообщении и затем шифрует его, находя соответствие с заданным криптотекстом. Даже если опубликованный алгоритм зашифрования является недетерминированным, вся процедура пока еще лежит в NP.

Задача криптоанализа лежит также в классе $Co\text{-}NP$. Если алгоритм зашифрования является детерминированным, то это очевидно, потому что он может действовать в точности, как и ранее: обнаружив, что данный кандидат исходного сообщения еще *не дает* заданного криптотекста. В общем случае, независимо от того, является ли алгоритм зашифрования детерминированным, мы докажем это следующим образом. Зададим тройку

$$(w, k, c) \ ,$$

где w является вариантом исходного сообщения, k — это открытый ключ зашифрования и c — криптотекст. Предположим, что тройка точна в случае, если w является исходным сообщением, дающим в результате c. Очевидно, что существует только один такой исходный текст, иначе расшифрование было бы неоднозначным.

Сначала наш алгоритм выдает возможный вариант исходного сообщения p, затем определяет (в недетерминированное полиномиальное

время), дает ли p в результате c согласно k. Только в случае положительного ответа алгоритм продолжает свою работу, сравнивая p и w по буквам. Если различие найдено, алгоритм допустим.

Мы рассмотрели задачу криптоанализа в обычном смысле: найти исходное сообщение, когда известен криптотекст и опубликован ключ. Можно показать, что *несколько аналогичных задач принадлежат пересечению* $NP \cap Co\text{-}NP$. В каждом случае мы полагаем, что заданы ключ зашифрования и криптотекст.

(i) Появляется ли заданное слово как префикс (соответственно суффикс) в исходном тексте?
(ii) Появляется ли заданное слово как подслово в исходном тексте?
(iii) Получается ли заданное слово только из букв в позициях 5, 10, 15,... исходного текста?

Таким образом *задача криптоанализа для криптосистем с открытым ключом лежит в пересечении* $NP \cap Co\text{-}NP$. Следовательно, если задача криптоанализа C NP-полна, то $NP = Co\text{-}NP$. Это можно показать с помощью следующего простого рассуждения. Рассмотрим любую задачу L из NP. Так как C NP-полна, то L за полиномиальное время сводится к C. Дополнение L за полиномиальное время сводится к дополнению C, которое, по нашему предположению, лежит в $Co\text{-}NP$. Поэтому L из $Co\text{-}NP$ и, следовательно, $NP \subset Co\text{-}NP$. По данному включению очевидно и обратное включение. Возьмем любую $L \in Co\text{-}NP$. Дополнение $L \in NP$, а также $Co\text{-}NP$. Из этого следует, что $L \in NP$.

Мы показали, что если задача криптоанализа для системы с открытым ключом NP-полна, то $NP = Co\text{-}NP$. Поэтому *очень маловероятно, что задача криптоанализа для систем с открытым ключом является* NP*-полной или принадлежит более высокому классу сложности*. Мы можем найти примеры, оптимальные с точки зрения теории сложности.

Пример 2.2 (из [Kar1]). Рассмотрим задачу выполнимости для РФПИ (регулярных формул пропозиционального исчисления — см. приложение А) с переменными из $X \cup Y$, где X и Y не пересекаются. Каждая такая регулярная формула строится из переменных с помощью логических связок $\vee, \&$ и $\neg$. Полагаем, что формулы могут принимать истинностные значения T и F.

Пусть α является набором значений для переменных из X, а p_0 и p_1 — две такие регулярные формулы, что p_0 принимает значение T и p_1 принимает значение F для каждого набора переменных из $X \cup Y$, в которых набором значений для переменных из X является α. Таким

образом, если α используется для X, то значения p_0 и p_1 не зависят от значений переменных из Y. Пара (p_0, p_1) образует открытый ключ зашифрования, тогда как α является секретной лазейкой.

В качестве иллюстрации рассмотрим $X = \{x_1, x_2\}$ и $Y = \{y_1, y_2\}$. Определим α следующим образом:

$$\alpha(x_1) = F \qquad \text{и} \qquad \alpha(x_2) = T \ .$$

Теперь можно выбрать

$$\begin{array}{lcl} p_0 & = & \neg y_1 \& y_2 \& x_2 \& (y_1 \vee x_1 \vee (\neg y_2 \& x_2)) \ , \\ p_1 & = & (y_2 \vee x_2) \& (y_1 \vee x_1 \vee (\neg y_1 \& x_2)) \ . \end{array}$$

Легко видеть, что, независимо от значений для y_1 и y_2, p_0 принимает значение F и p_1 — значение T для α.

Для зашифрования конкретного появления бита i в исходном тексте подставляем в p_i произвольные значения для переменных в Y и случайным образом преобразуем результирующую регулярную формулу (с переменными из X) согласно стандартным правилам пропозиционального исчисления (добавление и поглощение T и F, ассоциативность, коммутативность, дистрибутивность, идемпотентность). Если мы сопоставим значения F и T для y_1 и y_2 в нашей иллюстрации, то p_0 можно записать в виде

$$\neg F \& T \& x_2 \& (F \vee x_1 \vee (\neg T \& x_2)) \ .$$

Это выражение может быть преобразовано к $x_2 \& x_1$. В результате $x_2 \& x_1$ является одним из возможных шифрсообщений для бита 0.

Легальное расшифрование является быстрым, так как известно α. Используя NP-полноту задачи, можно получить следующий результат. Предположим, что мы можем консультироваться с *оракулом*, который по заданным открытому ключу и криптотексту сообщает нам бит, из которого получен криптотекст (оракулы будут обсуждаться более подробно в гл. 4). Тогда для каждого языка в пересечении NP и $Co\text{-}NP$ существует детерминированный полиномиальный алгоритм, использующий оракула для ответа на вопрос: принадлежит ли заданное слово языку. Этот результат означает, что криптоанализ любой системы с открытым ключом может быть сведен к криптоанализу системы, описанной выше. Таким образом, система оптимальна в том смысле, что любой криптоаналитический метод для ее вскрытия может быть использован при вскрытии любой другой криптосистемы с открытым ключом.

К несчастью, тот же результат может быть получен для следующей вырожденной системы. В открытом ключе (p_0, p_1) только одна из

формул p, скажем p_k, выполнима. Индекс k образует секретную лазейку. Появление бита i шифруется сопоставлением значений для переменных в p_i произвольным образом. Если результирующее значение для p_i есть T, то i шифруется как #, иначе i шифруется как i.

При легальном расшифровании мы просто отображаем # в k и оставляем 0 и 1. С другой стороны, криптоаналитик может найти значение # генерацией предположений до тех пор, пока p_0 или p_1 не станет истинной. Если формула p_k редко является истинной, то и # появляется в криптотексте редко. Таким образом, вырожденная система интуитивно очень слаба. Парадокс оптимальности системы объясняется тем фактом, что мы рассматривали худший случай в отличие от сложности в среднем.

□

Изложенные выше рассуждения и выводы мы проводили при условии, что "известен заданный криптотекст и открытый ключ зашифрования". Для условия "известен только ключ зашифрования" *задача криптоанализа принадлежит NP для любой криптосистемы с открытым ключом*. Довольно интересно, что система из примера 2.2 оптимальна также и при условии "известен только ключ зашифрования": задача криптоанализа является NP-полной.

Очевидно, что нет подобных верхних оценок для сложности криптоанализа классических криптосистем. По существу, это следует из того факта, что все хранится в секрете и легкость зашифрования и расшифрования для легальных пользователей никак не касается мира криптоаналитика.

Последнее несколько странное наблюдение может быть сделано с точки зрения теории сложности. Криптосистема с открытым ключом может всегда быть рассмотрена как последовательность пар (E_i, D_i), $i = 1, 2, \ldots$, где E_i — ключ зашифрования, а D_i — соответствующий ключ расшифрования. Оба ключа полностью определены числом i: они могут быть заданы некоторым устным описанием. Теперь вернемся к предварительному криптоанализу. После опубликования ключа зашифрования E_k генерируется последовательность (E_i, D_i), пока не будет найден правильный E_i (он устно совпадает с E_k). Это может привести к большому (вычислительно неосуществимому) времени работы. Однако это время является константой, *не зависящей от длины криптотекста*. С этой точки зрения сложность условия для криптоаналитика "известен криптотекст и ключ зашифрования" равна $n + c$, где c — константа! Конечно, с практической точки зрения это ни о чем не говорит, так как c огромна.

2.3. Очевидные преимущества открытых ключей

Преимущества криптографии с открытым ключом огромны, если идея может быть реализована без каких-либо вредных побочных эффектов. Наиболее полезное новшество, касающееся открытых ключей, связано с *управлением ключами*: с их выбором и рассылкой.

Рассмотрим любую классическую (т. е. симметрическую) криптосистему. Ключ зашифрования дает также и ключ расшифрования, следовательно, первый не может быть раскрыт. Это означает, что две легальные стороны (отправитель и получатель) договариваются *заранее* об алгоритме зашифрования. Это может случиться или при встрече между двумя сторонами, или при передаче ключа зашифрования по абсолютно секретному каналу.

При использовании криптосистемы с открытым ключом обе стороны не встречаются, они даже могут не знать друг друга и использовать любые виды связи. Это огромное преимущество, к примеру, в случае большого банка данных, где существует масса пользователей и один из них хочет связаться только с каким-то одним другим пользователем. Тогда он может проделать это путем самостоятельного извлечения информации из банка данных.

Можно сравнивать классические криптосистемы и криптосистемы с открытым ключом также и по *длине ключа*. Так как каждый ключ должен быть как-нибудь описан, то это описание является последовательностью букв некоторого алфавита (т.е. словом), длина которой говорит о длине ключа. Существует примечательное различие между классическими криптосистемами и криптосистемами с открытым ключом.

Рассмотрим сначала классическую криптосистему. Если ключ длиннее исходного сообщения, никакого действительного выигрыша не достигается. Так как ключ будет передаваться секретно, то можно передать исходное сообщение вместо ключа по этому секретному каналу. Конечно, в некоторых ситуациях ключ передается заранее до пересылки сообщений.

Рассмотрим теперь криптосистему с открытым ключом. Длина ключа зашифрования не относится к делу. Ключ открывается любым способом. Это означает, что и длина ключа расшифрования не относится к делу: получатель только хранит его в секретном месте.

Легкость в управлении ключами справедливо может признаваться главным преимуществом криптографии с открытым ключом. Теперь рассмотрим некоторые другие преимущества. Основные выводы будут также обсуждаться далее.

Одним из центральных оплотов компьютерных систем является файл *пароля*. Следующая пара может служить входом в такой файл.

имя: Джонсон пароль: KILLER

Если файл пароля раскрыт — случайно или нет — несанкционированным пользователем ("самозванцем"), то последний будет иметь свободный доступ, к примеру, к электронной почте мистера Джонсона. Мы полагаем здесь, что электронная почта не зашифрована и секретность, таким образом, достигается только паролями.

Предположим теперь, что для файла пароля используется *односторонняя* функция t. Вышеуказанный вход теперь выглядит следующим образом.

имя: JOHNSON пароль: KILLER функция: f_J

Здесь f_J — односторонняя функция. Идея в том, что KILLER является "открытым" паролем мистера Джонсона, тогда как только мистер Джонсон знает свой "секретный" пароль PURR, такой, что

$$f_J(\text{PURR}) = \text{KILLER}\ .$$

В действительности он "открывает" пароль KILLER после вычисления f_J(PURR).

Мистер Джонсон набирает секретный пароль PURR, после чего компьютер проверяет, дает или нет функция f_J, применяемая к PURR, правильный результат KILLER. Компьютер не хранит PURR ни в каком виде. Файл пароля может быть теперь просмотрен посторонними без потери секретности, потому что функция f_J не может быть обратимой.

Односторонние функции f не обязательно будут криптографическими: секретная лазейка для их обращения в этом случае не используется. Возможно даже иметь одну и ту же функцию для всех пользователей. Читатель может предположить, что такая функция, общая для всех, слабее, чем индивидуальные функции.

Важнейшее значение в описанном случае приобретает проблема *идентификации*. Как мы узнаем, что сообщение, передаваемое по каналу связи или в информационной системе, является достоверным? Как сгенерировать *электронную* или *цифровую подпись*? Сформулируем более точно цель нашего исследования.

Рассмотрим две стороны A и B, возможно с противоположными интересами. Например, сторонами могут быть банк и его клиенты, или две сверхдержавы. Когда A посылает сообщение B, оно подписывается таким образом, что стороны получают следующие два вида защиты.

1. Оба A и B защищены против сообщений, адресованных B, но переданных в информационную систему третьей стороной C, которая выдает себя за A.

2. A защищен против сообщений, подделанных B, который желает получить их от A, правильно подписанных.

Конечно, если B посылает сообщение A, то A и B будут меняться ролями в (2).

Можно наглядно представить (1) и (2), где B — американский агент в Москве, A — его босс в Вашингтоне, а C — русский агент. Важность требования (1) очевидна. Условие (2) требуется, к примеру, в случае, если B проводит некоторую операцию без всяких санкций от A. Операция может быть неудачной. Однако B претендует на действия согласно инструкциям, данным A, в правильно подписанном сообщении!

Условия (1) и (2) несколько противоречивы и, следовательно, трудно выполнить их одновременно. Согласно (1), B кое-что знает о подписи A. Согласно (2), B многого не знает о подписи A. Подчеркнем, что электронные подписи обычно изменяют все сообщение, в отличие от добавления в конец текста.

Если используется хорошая классическая криптосистема, то требование (1) может быть выполнено замечательным образом: A и B договариваются, что ключ зашифрования знают только они. Сообщение подписано самим зашифрованием, согласно ключу. Ключ и сама система будут меняться довольно часто. Когда C находит ключ, он может начинать посылать правильно подписанные сообщения.

Требование (2) выполнить труднее, потому что, как мы уже отмечали, B кое-что знает о способе A генерировать подпись и необходимо добиться невозможности для B генерации подписи A. Заметим также, что если мы связаны с большой сетью (такой, как сеть пользователей электронной почты), то непрактично использовать различные секретные методы подписи для каждой пары пользователей.

Если используется криптосистема с открытым ключом, то оба требования — (1) и (2) — выполнены, по крайней мере в принципе. Как и ранее, обозначим через E_A, E_B, ... (соответственно D_A, D_B, ...) ключи зашифрования (соответственно расшифрования), используемые сторонами A, B, ... Сначала A посылает сообщение w к B в виде $E_B(D_A(w))$. Тогда B может получить $D_A(w)$ с помощью его секретного ключа расшифрования D_B. Из $D_A(w)$ B может получить w с помощью открытого E_A. Заметим, что E_A и D_A взаимообратны.

Теперь оба условия (1) и (2) выполнены. Только A знает D_A, и,

следовательно, ни C, ни B не могут подделать подпись A. Это может быть только случайным, по крайней мере если исходные сообщения являются осмысленными фрагментами некоторого естественного языка. Тогда существует незначительная вероятность, что некоторый текст, полученный без помощи D_A из осмысленного исходного сообщения, будет переведен в некоторый осмысленный. По этой причине A также не может отрицать посылку сообщения к B.

Если важна только подпись (а не шифрсообщение), то достаточно, чтобы A послал B пару $(w, D_A(w))$. Требования (1) и (2) выполняются, как и ранее.

Основные процедуры идентификации, описанные выше, уязвимы, особенно это касается действий активных перехватчиков. Серьезность их попыток зависит от деталей, в частности от возможностей перехватчика посылать ложные сообщения в систему. Основные процедуры могут быть усилены применением *протокола*. Это означает, что посылка сообщения от A к B содержит несколько шагов связи между A и B. Первый связывается с B. В зависимости от содержания этой связи B связывается обратно с A. И так далее.

В общем протокол подразумевает обмен сообщениями. Число связывающихся сторон может быть также и больше двух. Для обеспечения протокола используется специальная криптосистема, чаще всего с открытым ключом. *Безопасность* протокола, как правило, означает защиту против пассивного или активного *перехватчика*, а часто и защиту против *обмана* некоторыми сторонами. В последнем случае протокол может обеспечить арбитражные процедуры, если стороны не соглашаются со строгой приверженностью протоколу. Протоколы не являются более надежными, чем применяемая криптосистема. Трудно доказать, что специфическая криптосистема обладает определенными свойствами безопасности. Также трудно доказать, что если используемая криптосистема удовлетворяет определенным условиям безопасности, то и протокол обладает такими же свойствами безопасности.

Многие из сделанных здесь выводов будут детализированы в гл. 6. Здесь мы кратко отметим проблемы и задачи, для которых применение протоколов является успешным.

Рукопожатие является более сложным, чем идентификация. Задача состоит в том, что A и B хотят установить надежный канал в определенной коммуникативной среде без всякого предшествующего обмена информацией. В нашем предыдущем примере американский агент в Москве и его босс в Вашингтоне заранее договорились, по крайней мере о следующем: как в принципе генерируется подпись и где находится доступ к открытым ключам (полагаем, что они используют основную процедуру, описанную выше). Это не так много и может

быть заключено в общие инструкции для пользователей информационной системы. Следовательно, ситуация является очень близкой к рукопожатию.

Очень часто под рукопожатием подразумевается, что стороны доверяют друг другу. Тогда требование (2) становится излишним.

Предположим, что *выборы* проходят с использованием компьютерной сети. Протокол делает невозможным голосование для незарегистрированных избирателей, хотя они могут быть легальными пользователями сети. Более того, избирательные бюллетени держатся в секрете, и публикуемые итоги выборов будут честными.

Некоторые новые типы секретных голосований также могут осуществляться с использованием подходящих протоколов. Такие протоколы открывают новые возможности для доверительной связи.

Некоторые члены совета могут иметь право на вето. Когда используется подходящий протокол, никто не знает, отрицательное решение основано на большинстве, или кто-либо использовал свое право на вето, или и то и другое вместе!

Рассмотрим специфический пример. Стороны $A, B, C_1, \ldots, C_n$ хотят сказать "да" или "нет" по некоторому вопросу. Все стороны могут голосовать "за" или "против". Такое голосование может быть представлено как возникающее в ООН, где A и B — сверхдержавы. Если суперголосующие не подают голоса, то решает большинство. Если по крайней мере один суперголосующий отдал голос, то обычные голоса не учитываются. В случае ничьей решением является "за". После голосования все стороны знают решение, но никто не знает, как оно принималось. Что повлияло — большинство, суперголосующий или и то и другое вместе? Конечно, возможно построить машину для голосования, удовлетворяющую этим требованиям. Но никто не будет верить такой машине: в ней может быть вмешательство в текущую информацию и в принятие ложного итога выборов.

В следующем примере предлагается специфический протокол.

Пример 2.3. Две стороны A и B хотят сыграть в покер по телефону без посредника, действующего в качестве беспристрастного судьи. Рассмотрим основной вариант игры, где раздаются по пять карт. Что касается большинства других вариантов, то протокол будет существенно тем же. Очевидно, что для A и B необходимо обменяться информацией в зашифрованном виде в порядке "сдачи" карт надлежащим образом.

Правильная сдача должна удовлетворять следующим требованиям:

1. Все наборы из пяти карт равновероятны.
2. Наборы карт игроков A и B не пересекаются.

3. Оба игрока знают карты на своей руке, но не имеют информации о картах на руках оппонента.
4. Для каждого из игроков имеется возможность обнаружения вероятного обмана другим игроком.

Теперь предложим протокол. Используется классическая или с открытым ключом криптосистема. Однако ни алгоритмы зашифрования E_A и E_B, ни алгоритмы расшифрования D_A, D_B не раскрываются. Предполагается *коммутативность*, т. е. в любой композиции E и D взаимный порядок не важен. Перед действительной игрой A и B соглашаются об именах $w_1, \ldots, w_{52}$ 52 карт. Имена выбираются таким образом, чтобы криптосистема была применимой. К примеру, если E_A и E_B оперируют целыми числами определенной разрядности, то каждое w_i должно быть целым числом этой разрядности.

Теперь мы готовы описать протокол. A действует как сдающий, но роли A и B могут меняться. Протокол содержит следующие четыре шага.

Шаг 1: B перемешивает карты, зашифровывает их, используя E_B, и сообщает результат A. Это означает, что B говорит A части $E_B(w_1), \ldots, E_B(w_{52})$ в случайно выбранном порядке.

Шаг 2: A выбирает пять $E_B(w_i)$ и сообщает их B. Эти части являются картами B.

Шаг 3: A выбирает *другие* пять частей $E_B(w_i)$, зашифровывает их с помощью E_A и сообщает результат B.

Шаг 4: После получения пяти частей в виде $E_A(E_B(w_i))$ в шаге 3 B применяет к ним D_B и сообщает результат A. Эти пять частей представляют карты A.

Теперь рассмотрим, как выполняются требования (1)–(4). Очевидно, что оба игрока знают свои собственные карты. В частности, A получает на шаге 4 пять частей в виде $D_B(E_A(E_B(w_i))$. В силу коммутативности

$$D_B(E_A(E_B(w_i)) = E_A(D_B(E_B(w_i)) = E_A(w_i) \ ,$$

и, следовательно, A нужно только использовать D_A. Карты на обеих руках не пересекаются: B может немедленно проверить, что части, данные в шаге 3, отличаются от частей, данных в шаге 2.

Нет убедительных доказательств, которые можно представить относительно требований (1)–(4). Все во многом зависит от выбора односторонних функций E. К примеру, может оказаться, что невозможно найти w_i на основе $E_B(w_i)$, однако может быть извлечена некоторая частичная информация. Например, если w_i — последовательность битов, то последний бит может быть найден из $E_B(w_i)$. Такая частичная

информация может сказать A, что все тузы образуют поднабор с определенным свойством в $E_B(w_1), \ldots, E_B(w_{52})$. Теперь A будет уверенно выбирать карты для B вне этого поднабора, а для себя — только карты из этого поднабора. В этом случае (1) и вторая часть (3) будут нарушены.

Криптосистема не может быть криптосистемой с открытым ключом в обычном смысле. A может просто вычислить все значения $E_B(w_i)$ и соответственно раздать карты: хорошие карты для B, но еще лучшие — для себя!

Некоторые выводы из этого примера являются общими и будут обсуждаться далее. Действительно, криптосистемы с открытым ключом никогда не могут иметь небольшое пространство исходных текстов, как в данном примере — 52 исходных сообщения. Тогда все они могут быть зашифрованы с помощью открытого ключа и расшифрование сводится к поиску среди всех результирующих криптотекстов.

Возможность получения частичной информации является также одним из центральных моментов в криптографии с открытым ключом. Для некоторых криптосистем, таких, как RSA, показано, что если может быть получена частичная информация, то и вся система может быть вскрыта. Это означает, что если вы убеждены в надежности криптосистемы, то вы также знаете, что система не имеет просачивания частичной информации.

□

Завершая эту главу, обратим внимание на три проблемы, которые требуют криптографических протоколов для их решения. Протоколы, созданные для этих проблем, часто используются как часть протокола для более сложной задачи. Так, протокол, представленный в [GM] для задачи из примера 2.3, использует подбрасывание монеты.

A и B хотят *бросить жребий по телефону* без участия беспристрастного судьи. Как всегда, обе стороны в состоянии проверить, что другая сторона не обманывает. После этого результат подбрасывания монеты может использоваться для некоторых других целей.

Забывчивая передача — A передает секрет B с вероятностью $1/2$. После завершения протокола B знает, успешно или нет был передан секрет, а A этого не знает.

Две или более сторон хотят сравнить *часть их секретов*, но не хотят при этом раскрывать свои секреты целиком. К примеру, два человека хотят определить, кто из них старше, не зная ничего о возрасте друг друга. После использования протокола оба знают, кто из них старше, но никто не знает, насколько.

Глава 3

Рюкзачные системы

3.1. Строим секретную лазейку

Криптосистемы с открытым ключом, основанные на задаче о рюкзаке, уже кратко обсуждались в примере 2.1 гл. 2. Там также указывалось, что рюкзачные системы очень удобны для иллюстрации основных идей криптографии с открытым ключом. Они являются достаточно гибкими и позволяют для преодоления криптографических слабостей вводить новые варианты.

В этой и последующих главах математический аппарат будет использоваться в большей степени, чем в гл. 1 и 2. Все необходимые понятия приведены в приложениях, хотя с основами теории можно также познакомиться и без погружения в математические детали.

В этом параграфе более подробно, чем в примере 2.1, представлена основная рюкзачная система. Криптоаналитический метод Шамира описывается в параграфе 3.2. В параграфе 3.3 развивается общая теория достижимости, применяемая как к простым, так и составным рюкзакам. Интересные варианты рюкзачных систем представлены в параграфе 3.4. В заключительном параграфе 3.5 рассматриваются системы, основанные на плотных рюкзаках.

Теперь мы готовы перейти к определениям. *Рюкзачный вектор* $A = (a_1, \ldots, a_n)$ — это упорядоченный набор из n, $n \geq 3$, различных натуральных чисел a_i. *Входом задачи о рюкзаке* называем пару (A, α), где A — рюкзачный вектор, а α — натуральное число. *Решением* для входа (A, α) будет такое подмножество из A, сумма элементов которого равняется α. (Поскольку мы говорим о подмножестве, каждое

a_i появляется в сумме самое большее один раз.) Задачу о рюкзаке иногда называют также задачей о сумме размеров.

В наиболее известном варианте задачи о рюкзаке требуется выяснить, обладает или нет данный вход (A, α) решением. В варианте, используемом в криптографии, нужно для данного входа (A, α) построить решение, зная, что такое решение существует. Оба эти варианта являются NP-полными. Имеются также варианты этой задачи, которые не лежат даже в классе NP.

Рюкзачный вектор A используется для зашифрования блока C из n двоичных символов путем суммирования тех компонент A, для которых в соответствующих позициях C стоит единица. Если эту сумму обозначить через α, то тогда расшифрование равносильно нахождению C по α или по A и α, если мы имеем дело с криптосистемой с открытым ключом. Второй вариант есть в точности криптографический вариант задачи о рюкзаке.

В эквивалентной форме, мы можем рассматривать C как двоичный вектор-столбец. Тогда α равно произведению AC.

Для иллюстрации положим $n = 6$ и $A = (3, 41, 5, 1, 21, 10)$. Тогда двоичные блоки $(1, 1, 0, 0, 1, 0)$ и $(1, 0, 1, 1, 0, 1)$ шифруются как 65 и 19 соответственно. Для данного A все криптотексты есть числа ≤ 81 и не более одного исходного текста соответствует каждому криптотексту.

В случае $A = (14, 28, 56, 82, 90, 132, 197, 284, 341, 455)$ криптотекст $\alpha = 55$ соответствует ровно трем исходным текстам

$$(1, 1, 0, 0, 1, 0, 0, 1, 0), \quad (0, 1, 1, 0, 1, 0, 0, 0, 1, 0), \quad (1, 0, 0, 1, 1, 1, 1, 0, 0, 0) \ .$$

Это сразу ясно, если начать читать A справа налево. Например, 455 не может входить в решение, так как невозможно выразить $60 = 515 - 455$ в виде суммы и т.д. Аналогично рассуждая, можно показать, что криптотекст $\alpha = 516$ не имеет соответствующего исходного текста. В этом случае легко видеть, что не одно из четырех последних чисел из A не может входить в сумму, тогда как сумма остальных чисел слишком мала. Для $\alpha = 517$ единственным соответствующим исходным текстом будет $(1, 1, 1, 0, 1, 1, 1, 0, 0, 0)$. Примеры такого рода иллюстрируют тот очевидный факт, что криптоанализ для некоторых входов задачи о рюкзаке может быть легким.

Поскольку желательна однозначность расшифрования, рюкзачные векторы A должны обладать таким свойством, что и для каждого α, все входы (A, α) обладают не более чем одним решением. Такие рюкзачные векторы A будем называть в дальнейшем *инъективными*. Этот термин очень естествен, поскольку инъективность A означает, что порожденная вектором A функция, определяемая в примере 2.1, является

инъективной. Среди рассмотренных выше двух векторов первый является инъективным, в то время как второй — нет.

Для некоторых векторов A все входы (A, α) являются легкорешаемыми. Мы уже видели в примере 2.1, что сверхрастущие векторы обладают этим свойством. Опираясь на такие векторы, можно очевидным образом построить двустороннюю криптосистему: оба и отправитель сообщения, и его получатель знают вектор A. С другой стороны, если вектор B раскрыт как ключ зашифрования, то легальный получатель должен владеть некоторой секретной информацией для преобразования как B, так и самого криптотекста в легкорешаемый вход задачи о рюкзаке. Мы уже показывали в примере 2.1, как это может быть сделано с использованием сверхрастущих векторов. Эта конструкция будет описана сейчас несколько более подробно.

Рюкзачный вектор $A = (a_1, \ldots, a_n)$ называем *возрастающим* (соответственно *сверхрастущим*), если и только если

$$a_j > a_{j-1} \quad \left(\text{соответственно } a_j > \sum_{i=1}^{j-1} a_i\right)$$

выполняется для всех $j = 2, \ldots, n$. Ясно, что любой сверхрастущий вектор является возрастающим. Для вектора A мы определим

$$\max A = \max(a_j | 1 \leq j \leq n) \ .$$

Пусть x неотрицательное число. Обозначим через $[x]$ целую часть x, т. е. наибольшее целое $\leq x$.

Для целых x и $m \geq 2$ обозначаем через $(x, \mathrm{mod} m)$ *наименьший неотрицательный остаток* от деления x на m. Легко видеть, что

$$(x, \mathrm{mod} m) = x - [x/m] \cdot m \ .$$

Это равенство будет часто, особенно в параграфе 3.3, записываться в виде

$$x = (x, \mathrm{mod} m) + [x/m] \cdot m \ .$$

Определим теперь два варианта понятия *модульного умножения*. Рассмотрим рюкзачный вектор A, целое число $m > \max A$ и натуральное $t < m$ такое, что наибольший общий делитель $(t, m) = 1$. Если $B = (b_1, \ldots, b_n)$ такой вектор, что

$$b_i = (ta_i, \mathrm{mod} m), \quad \text{для } i = 1, \ldots, n \ ,$$

то говорят, что вектор B получен из A с помощью *модульного умножения* относительно *модуля* m и *множителя* t или, короче, относительно

пары (m,t). Условие $(t,m)=1$ гарантирует существование обратного числа $t^{-1}=u$, такого, что

$$tu \equiv 1 \pmod{m}$$

и $1 \le u < m$. Это означает, что также и обратно A получается из B модульным умножением относительно m и u. (Ясно, что $m > \max B$, так как каждое b_i берется по $\mathrm{mod} m$.)

Если вышеуказанное условие $m > \max A$ заменяется более сильным условием $m > \sum_{i=1}^{n} a_i$, то говорят, что B получается из A *сильным модульным умножением* относительно m и t. Заметим, что сейчас мы не можем заключить, что A получается из B сильным модульным умножением относительно m и u, так как неравенство $m > \sum_{i=1}^{n} b_i$ не обязательно выполняется. Конечно, A получается из B модульным умножением относительно m и u.

Создатель криптосистемы выбирает теперь A, t, m, B так, что вектор A является сверхрастущим, а B получается из A сильным модульным умножением относительно m и t. Вектор B раскрывается как ключ зашифрования и двоичные блоки длины n посылаются к проектировщику как числа β, полученные с помощью вектора B, как было описано выше. Перехватчик сообщений должен решать задачу о рюкзаке для входа (B,β). Создатель же криптосистемы вычисляет $\alpha = (u\beta, \mathrm{mod} m)$ и решает задачу о рюкзаке для входа (A,α). Почему все это работает, показывает следующая лемма.

Лемма 3.1 *Предположим, что $A=(a_1,\ldots,a_n)$ сверхрастущий вектор и вектор B получен из A сильным модульным умножением относительно m и t. Предположим далее, что $u \equiv t^{-1} \pmod{m}$, β — произвольное натуральное число и $\alpha = (u\beta, \mathrm{mod} m)$. Тогда справедливы следующие утверждения. (i) Задача о рюкзаке (A,α) разрешима за линейное время. Если решение существует, то оно единственно. (ii) Задача о рюкзаке (B,β) имеет не более одного решения. (iii) Если существует решение для входа (B,β), то оно совпадает с единственным решением для входа (A,α).*

Доказательство. (i) В примере 2.1 было показано, что любая задача о рюкзаке со сверхрастущим вектором A может быть разрешена за линейное время путем однократного считывания A справа налево. Этот метод показывает также, что в задаче может быть самое большее одно решение. (ii) и (iii) Предположим, что двоичный вектор D длины n есть решение в задаче (B,β), т. е., $BD=\beta$. Следовательно,

$$\alpha \equiv u\beta = uBD \equiv u(tA)D \equiv AD \pmod{m}\ .$$

Поскольку m превосходит сумму компонент вектора A, имеем $AD < m$. Поскольку также $\alpha < m$, по определению α заключаем, что $\alpha = AD$. Таким образом, D совпадает с единственным решением в задаче (A, α). Это показывает (iii). Поскольку мы рассматривали произвольное решение в задаче (B, β) и показали, что оно совпадает с единственным решением задачи (A, α), то мы также установили и (ii).

□

В приложении леммы 3.1 к криптографии мы знаем, что задача (B, β) заведомо имеет решение: число β было вычислено способом, который это гарантирует.

Пример 3.1. В нашем первом примере все еще можно обойтись карманным калькулятором. Пусть $n = 10$ и рассмотрим сверхрастущий вектор

$$A = (103, 107, 211, 430, 863, 1718, 3449, 6907, 13807, 27610) .$$

Выберем модуль $m = 55207$, который больше (на два) суммы компонент A. Выберем далее множитель $t = 25236$. Тогда $(t, m) = 1$ и $t^{-1} = u = 1061$. Действительно,

$$1061 \cdot 25236 - 1 = 485 \cdot 55207 .$$

В результате сильного модульного умножения теперь получаем вектор

$$B = (4579, 50316, 24924, 30908, 27110, 17953, 32732, 16553, 22075, 53620) .$$

Например,

$$25236 \cdot 103 = 4579 + 47 \cdot 55207 \text{ и } 1061 \cdot 4579 = 103 + 88 \cdot 55207 ,$$

$$25236 \cdot 1718 = 17953 + 785 \cdot 55207 \text{ и } 1061 \cdot 17953 = 1718 + 345 \cdot 55207 ,$$

$$25236 \cdot 27610 = 53620 + 12620 \cdot 55207 \text{ и } 1061 \cdot 53620 = 27610 + 1030 \cdot 55207 .$$

Вектор B есть открытый ключ шифрования, в то время как A, t, u, m составляют секретную лазейку. Конечно, зная m и t или u, можно вычислить остальные величины.

Применим сейчас открытый ключ B и зашифруем исходное сообщение IN FINLAND CHILDREN USED TO BE BORN IN SAUNA EVEN TODAY INFANT MORTALITY IS IN FINLAND LOWEST IN THE WORLD. Вначале используем цифровое кодирование, при котором пробел получает значение 0, а буквы A–Z — значения 1–26. Цифровые коды выражаем двоичными наборами. Полный список двоичных наборов уже

приводился в примере 2.1. Поскольку вектор B может быть использован при зашифровании двоичных блоков длины 10, наш исходный текст следует разбить на блоки, состоящие из двух букв. В нижеследующей таблице приводится вначале блок исходного текста, затем двоичный код и, наконец, шифр блока в виде десятеричного числа. Криптотекст состоит из 53 таким образом полученных чисел, записанных одно за другим так, чтобы отдельные числа можно было бы различить.

IN	01001	01110	148786
F	00000	00110	38628
IN	01001	01110	148786
LA	01100	00001	128860
ND	01110	00100	122701
C	00000	00011	75695
HI	01000	01001	136668
LD	01100	00100	91793
RE	10010	00101	105660
N	01110	00000	106148
US	10101	10011	150261
ED	00101	00100	68587
T	00000	10100	34506
O	01111	00000	133258
BE	00010	00101	101081
B	00000	00010	22075
OR	01111	10010	173286
N	01110	00000	106148
IN	01001	01110	148786
S	00000	10011	93648
AU	00001	10101	115236
NA	01110	00001	159768
E	00000	00101	70173
VE	10110	00101	130584
N	01110	00000	106148
TO	10100	01111	154483
DA	00100	00001	78544
Y	11001	00000	82005
IN	01001	01110	148786
FA	00110	00001	109452
NT	01110	10100	140654
M	00000	01101	102905
OR	01111	10010	173286

TA	10100	00001	83123
LI	01100	01001	161592
TY	10100	11001	133808
I	00000	01001	86352
S	10011	00000	62597
IN	01001	01110	148786
F	00000	00110	38628
IN	01001	01110	148786
LA	01100	00001	128860
ND	01110	00100	122701
L	00000	01100	49285
OW	01111	10111	243459
ES	00101	10011	1456872
T	10100	00000	29503
IN	01001	01110	148786
T	00000	10100	34506
HE	01000	00101	120489
W	00000	10111	110201
OR	01111	10010	173286
LD	01100	00100	91793

Дешифруем первое число 148786. Отметим сначала, что

$$1061 \cdot 148786 = 2859 \cdot 55207 + 25133 \ .$$

Рассмотрим задачу о рюкзаке $(A, 25133)$. Решение получается просмотром A справа налево. Как только число в левом столбце становится не меньше текущей просматриваемой компоненты A, записываем единицу, и новое число в левом столбце получается вычитанием компоненты из предыдущего числа. В противном случае записываем символ 0 и число слева не меняется. Результат этих действий может быть записан следующим образом:

Число	Компонента A	Символ
25133	27610	0
25133	13807	1
11326	6907	1
4419	3449	1
970	1718	0
970	863	1
107	430	0
107	211	0
107	107	1
0	103	0

Исходный двоичный вектор, из которого получается блок IN, считывается из правой колонки снизу вверх. При дешифровке второго числа 38628 вначале получаем 20714, с которым поступаем аналогично, и т.д.

Здесь необходимо сделать одно замечание. Предположим, что мы пытаемся действовать в обратном порядке. Рассмотрим, например, блок OR, встречающийся три раза. Шифруя его с помощью вектора A, получим 7665. Но ясно, что пара $(B, 7665)$ не обладает решением. Простое объяснение этому состоит в том, что мы не можем вывести обычное равенство чисел из их равенства по модулю (как это было сделано в доказательстве леммы 3.1), потому что m меньше, чем сумма компонент вектора B. Действительно,

$$7665 \equiv 173286 \pmod{55207} ,$$

и нам следовало бы оперировать с числом 173286.

Наш второй пример уже чересчур громоздок для карманного калькулятора, но тем не менее слишком мал для реального зашифрования. Реальные же примеры, весьма вероятно, окажутся совсем сложными для восприятия. Приводимые ниже вычисления, так же как и последняя иллюстрация в примере 4.1, выполнены Киммо Кари.

Положим $n = 20$. Выберем модуль и сомножитель

$$m = 53939986 \text{ и } t = 54377 ,$$

для которых $t^{-1} = u = 17521047$. Определим сверхрастущий вектор A:

$$\begin{array}{rrrr} a_1 &=& 101 & a_{11} = 52676 \\ a_2 &=& 102 & a_{12} = 105352 \\ a_3 &=& 206 & a_{13} = 210703 \\ a_4 &=& 412 & a_{14} = 421407 \\ a_5 &=& 823 & a_{15} = 842812 \\ a_6 &=& 1647 & a_{16} = 1685624 \\ a_7 &=& 3292 & a_{17} = 3371249 \\ a_8 &=& 6584 & a_{18} = 6742497 \\ a_9 &=& 13169 & a_{19} = 13484996 \\ a_{10} &=& 26337 & a_{20} = 26969992 \end{array}$$

Сильное модульное умножение дает следующий открытый вектор B:

$$\begin{array}{ll} b_1 = 5492077 & b_{11} = 5543594 \\ b_2 = 5546454 & b_{12} = 11087188 \\ b_3 = 11201662 & b_{13} = 22119999 \\ b_4 = 22403324 & b_{14} = 44294375 \\ b_5 = 44752271 & b_{15} = 34540010 \\ b_6 = 35618933 & b_{16} = 15140034 \\ b_7 = 17189126 & b_{17} = 30334445 \\ b_8 = 34378252 & b_{18} = 6674527 \\ b_9 = 14870895 & b_{19} = 13457808 \\ b_{10} = 29687413 & b_{20} = 26915616 \end{array}$$

Зашифруем следующий текст о сауне: IF YOUR FEET CARRY YOU TO SAUNA THEY SURELY CARRY YOU BACK HONE IF SAUNA ALCOHOL AND TAR DO NOT CURE YOUR DISEASE IT MUST BE FATAL. Как и ранее, пробел кодируется 0, а буквы A–Z — числами 1–26. Пять битов требуется для двоичной записи каждого числа. Поскольку $n = 20$, четыре буквы текста зашифровываются одновременно. Кодирование блоков по 20 битов выглядит следующим образом:

IF Y	01001	00110	00000	11001
OUR	01111	10101	10010	00000
FEET	00110	00101	00101	10100
CAR	00000	00011	00001	10010
RY Y	10010	11001	00000	11001
OU T	01111	10101	00000	10100
O SA	01111	00000	10011	00001
UNA	10101	01110	00001	00000
THEY	10100	01000	00101	11001
SUR	00000	10011	10101	10010
ELY	00101	01100	11001	00000
CARR	00011	00001	10010	10010
Y YO	11001	00000	11001	01111
U BA	10101	00000	00010	00001
CK H	00011	01011	00000	01000
OME	01111	01101	00101	00000
IF S	01001	00110	00000	10011
AUNA	00001	10101	01110	00001
ALC	00000	00001	01100	00011
OHOL	01111	01000	01111	01100
AND	00000	00001	01110	00100

TAR	00000	10100	00001	10010
DO	00000	00100	01111	00000
NOT	01110	01111	10100	00000
CURE	00011	10101	10010	00101
YOU	00000	11001	01111	10101
R DI	10010	00000	00100	01001
SEAS	10011	00101	00001	10011
E IT	00101	00000	01001	10100
MUS	00000	01101	10101	10011
T BE	10100	00000	00010	00101
FAT	00000	00110	00001	10100
AL	00001	01100	00000	00000

Криптотекст состоит теперь из следующих чисел (см. замечание к шифровке в конце этого примера):

1 3 4 4 5 2 7 0 1
1 7 4 6 8 6 9 5 6
1 9 0 6 2 3 6 8 3
1 0 2 5 4 8 4 4 0
2 1 4 2 7 5 7 1 2
1 8 3 7 6 4 3 5 0
1 5 3 5 9 4 3 6 3
1 6 1 8 5 0 6 7 2
2 2 0 5 2 9 3 7 5
2 0 1 1 5 4 1 1 5
1 6 8 4 0 6 1 7 6
1 4 8 1 9 3 3 3 7
1 8 0 3 3 4 2 1 6
7 1 4 1 1 3 8 0
1 2 8 8 0 2 9 6 0
2 0 7 5 6 1 9 6 7
1 1 7 5 9 5 8 3 1
1 4 9 2 7 3 9 8 7
6 5 8 3 1 2 7 2
2 4 5 5 6 3 3 8 1
8 3 1 8 3 5 2 9
1 4 2 5 7 7 6 6 7
1 2 4 1 7 7 2 0 5
1 9 7 5 7 7 6 0 1
1 7 1 2 4 8 3 6 0
2 4 7 8 8 1 1 9 5
1 1 9 5 2 3 7 1 4

1 9 1 4 6 3 4 2 3
1 2 8 2 5 8 3 2 2
2 2 7 4 3 3 3 6 8
6 7 4 7 3 0 0 8
1 2 4 7 8 0 0 5 3
8 1 5 5 4 4 0 8

Легальный получатель сообщения умножает эти числа на $u \pmod m$ и возвращается к сверхрастущему вектору A. Например, умножение первого числа дает 15488011. Решая относительно A, так же как и в первом примере, получаем:

Число	Компонента A	Бит
15488011	26969992	0
15488011	13484996	1
2003015	6742497	0
2003015	3371249	0
2003015	1685624	1
317391	842812	0
317391	421407	0
317391	210703	1
106688	105352	1
1336	52676	0
1336	26337	0
1336	13169	0
1336	6584	0
1336	3292	0
1336	1647	0
1336	823	1
513	412	1
101	206	0
101	102	0
101	101	1

Процедура зашифрования в этом примере была необычной: порядок компонент вектора B был обратным. Так, для получения первого зашифрованного числа 134452701, была образована сумма $b_{19}+b_{16}+b_{13}+b_{12}+$ $+b_5+b_4+b_1$. Эта процедура предшествовала анализу A справа налево в вышеприведенной таблице. Однако в дальнейшем такая процедура не будет повторяться, поскольку она неестественна с точки зрения умножения векторов.

□

3.2. Как искать секретную лазейку

Перед нами стоит следующая криптоаналитическая задача. Нам известен рюкзачный вектор $B = (b_1, \ldots, b_n)$. Вектор B используется как открытый ключ зашифрования описанным выше способом. Также известно, что вектор B получен из сверхрастущего вектора A сильным модульным умножением относительно модуля m и множителя t. Вектор A и числа m и t нам неизвестны и мы хотим найти их. Что интересует нас больше всего, это найти m и $t^{-1} \equiv u \pmod{m}$. Зная m и u, можно сразу же вычислить A и расшифровать любой криптотекст. Вычисление u по t или, наоборот, t по u равносильно применению алгоритма Евклида и может быть выполнено быстро.

Здесь мы имеем условие криптоанализа *"известен только ключ зашифрования"*. Это часто означает, что в распоряжении криптоаналитика имеется достаточно времени, так как анализ системы может быть выполнен до того, как будут посланы важные криптотексты.

В этом параграфе обсуждается криптоаналитический подход А. Шамира. Получающийся в результате алгоритм является полиномиальным. Однако следует подчеркнуть, что классификация криптосистем на плохие и хорошие будет значительно упрощена, если внимание будет обращено только на условие *существования* полиномиального алгоритма для криптоанализа. Степень полинома весьма важна в криптографии. Более того, как уже было подчеркнуто, рюкзачные системы достаточно гибки для порождения модификаций, которые могут устоять против известных криптоаналитических методов.

Говоря о том, что алгоритм работает полиномиальное время, следует быть аккуратным в определении *размера* входа B, относительно которого алгоритм является полиномиальным. Мы должны рассматривать семейство рюкзачных векторов B с размерами, возрастающими до бесконечности. Имеются два параметра, дающих вклад в размер вектора B: число компонент n и размеры индивидуальных компонент b_i. Если любой из этих параметров ограничить сверху, то возникающая задача о рюкзаке тривиально разрешима за полиномиальное время.

Действительно, если любое b_i в каждом рассматриваемом векторе не превосходит некоторой константы C, то общее число таких целочисленных векторов будет конечным и, таким образом, имеется некоторая фиксированная временная граница, такая, что любая из рассматриваемых задач о рюкзаке может быть решена за линейное время с коэффициентом 2^C.

Обычно в качестве размера выбирается число компонет n и определяются границы для компонент в зависимости от n. Следует подчеркнуть, что ограничения для компонент такого рода с математической

точки зрения являются искусственными и ограничивают общность задачи, так как лишь незначительное число входов задачи оказывается внутри этих границ. Это также ясно и с точки зрения теории, развиваемой в параграфе 3.3.

В [Sh2] делаются следующие ограничения. Фиксируется константа пропорциональности $d > 1$. Затем выбирается модуль, состоящий из dn двоичных разрядов. Компоненты a_i, $1 \leq i \leq n$, сверхрастущего вектора A состоят из $dn - 1 - n + i$ битов. Если d не является целым, dn заменяется на $[dn]$. Старший разряд в каждой компоненте равен единице. Это гарантирует, что A всегда будет сверхрастущим и выбрано m, превосходящее по величине сумму компонент A. В статье [МеН] было рекомендовано выбирать $n = 100$ и $d = 2$. Это означает, что m состоит из 200 двоичных разрядов, а число разрядов в компонентах $a_1, \ldots, a_{100}$ возрастает от 100 до 199.

Следует отметить, что при построении алгоритма не обязательно искать обратный множитель u и тот модуль m, которые действительно использовались создателем криптосистемы. Нас устроит любая пара (u, m) при условии, что u и m удовлетворяют ограничениям, накладываемым на модульное умножение в отношении вектора B, что вектор A, возникающий в результате такого модульного умножения, является сверхрастущим и что m превосходит сумму компонент A. (Отсюда следует, что B получается из A сильным модульным умножением относительно m и $u^{-1} = t$.) Также пары (u, m) будем называть *секретными парами*. Как только мы найдем хотя бы одну секретную пару, мы сможем применить лемму 3.1 и начать расшифровку, используя полученный сверхрастущий вектор. И это совершенно не зависит от того, будут ли найденная секретная пара и сверхрастущий вектор теми, что реально использовались создателем криптосистемы. С другой стороны, существование по крайней мере одной такой секретной пары гарантируется тем, что создатель криптосистемы одну такую пару использовал. (Используя терминологию следующего параграфа, мы знаем априори, что данный рюкзачный вектор B является супердостижимым.)

Для того чтобы найти секретную пару (u, m), рассмотрим вначале графики функций $b_i u \pmod m$ для всех значений $i = 1, \ldots, n$. График функции $b_i u \pmod m$ состоит из прямолинейных отрезков, а значения $u = pm/b_i, p = 1, 2, \ldots$, являются точками разрыва. Так, на рис.3.1 показан график функции $b_i u \pmod m$, который имеет пилообразную форму. Такая пилообразная кривая рассматривается для всех $i = 1, \ldots, n$.

Напомним, что $(b_1 u, \mathrm{mod} m) = a_1$, где u является не переменной, а обратным множителем, который ищется. Поскольку a_1 является первой компонентой сверхрастущего вектора и m превосходит сумму всех

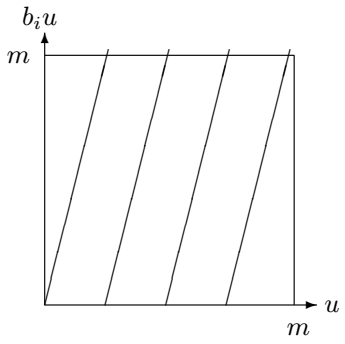


Рис. 3.1.

компонент, то a_1 должно быть очень мало по сравнению с m. Отсюда следует, что значение u из секретной пары должно быть близко к некоторому минимуму функции b_1. Более точная оценка того, насколько близко оно должно быть, включает в себя ряд ограничений (подобно доказанному выше) о значениях a_1 и m, а также об ожидаемом значении b_1. Обычно исходят из того, что b_i/a_i является большим для малых значений i. Однако создатель криптосистемы может специально позаботиться о том, чтобы выполнялось $b_i/a_i < 1$ для некоторых значений i. В этом случае некоторые расстояния будут существенно больше предполагаемых, что приводит к серьезным трудностям для криптоаналитика.

Аналогично рассуждая, приходим к тому, что значение u из секретной пары должно быть близко к некоторому минимуму функции b_2. Это влечет (по неравенству треугольника), что какие-то два минимума функций b_1 и b_2 должны быть близки друг к другу.

Действуя таким же образом, можно рассмотреть и другие пилообразные кривые. Тот факт, что значение u из секретной пары близко к некоторому минимуму каждой такой кривой, означает, что все эти минимумы близки друг к другу. Таким образом, вместо того чтобы пытаться найти само n, мы можем пытаться найти “точки накопления” минимумов наших пилообразных кривых. Это равносильно построению некоторого малого интервала, содержащего минимумы каждой из взятых пилообразных кривых. Исходя из этого интервала, мы также найдем и значение u из секретной пары. Эвристическими подсчетами (см. [Sh2]) можно показать, что для значения $d = 2$ константы про-

порциональности достаточно проанализировать только первые четыре пилообразные кривые, чтобы получить приемлемое (не слишком большое) множество точек накопления минимумов. Любая точка накопления минимумов для всех кривых будет находиться среди точек накопления, построенных только для минимумов первых четырех кривых.

Перейдем теперь к вопросу о том, как выразить эти идеи в виде неравенств. Первое препятствие состоит здесь в том, что мы не знаем значения модуля m из секретной пары. Это препятствие легко преодолимо. Сократим размер рисунка, изменяя масштаб, так, чтобы m стало равным 1. Другими словами, все длины делятся на m. Эта операция не влияет на расположение интересующих нас точек накопления. Например, если некоторый b_i-минимум был около седьмого b_3-минимума, то это же, конечно, справедливо и после изменения масштаба.

Алгоритм нахождения секретной пары состоит из двух частей. В первой части отыскиваются кандидаты для целого p, такие, что p-й минимум кривой b_1 является точкой накопления, которую мы ищем. Во второй части алгоритм проверяет кандидатов один за другим. Одна из этих проверок должна обязательно быть успешной, так как значение u из секретной пары, которую использовал создатель криптосистемы, определяет одну из точек накопления.

Следует принять некоторые меры предосторожности, так как первая часть алгоритма может породить слишком много (по сравнению с размером исходной задачи) кандидатов для p. Поэтому зафиксируем заранее параметр r — максимальное число разрешенных кандидатов. Если первая часть алгоритма породит $r+1$ кандидата для p, алгоритм прерывает свою работу. Таким образом, алгоритм будет стохастическим с некоторой малой вероятностью неудачи.

С другой стороны, мы не обязаны в первой части рассматривать все компоненты $b_2, \ldots, b_n$, а можем заранее зафиксировать значение другого параметра $s < n$ и рассматривать только компоненты $b_2, \ldots, b_s$. Другими словами, первая часть алгоритма порождает значения p, такие, что p-й минимум кривой b_1 располагается около некоторого минимума кривой b_i для $i = 2, \ldots, s$. Таким образом, значения $i > s$ в первой части алгоритма не рассматриваются вообще, и весьма вероятно, что будут порождены совершенно неверные значения p. Однако во второй части алгоритма проверяются все значения $i, 2 \le i \le n$. Кандидат p отбрасывается, если для некоторого i нет минимума кривой b_i, лежащего около p-го минимума кривой b_1. Мы уже указывали, что $s = 4$ во многих случаях является разумным выбором.

Рассмотрим первую часть алгоритма более детально. Координата u p-го минимума кривой b_1 есть p/b_1. (Напомним, что мы уменьшили рисунок таким образом, что модуль равняется 1.) Следовательно, условие,

что некоторый минимум кривой b_2 лежит около p-го минимума кривой b_1, может быть выражено как

$$-\epsilon < \frac{p}{b_1} - \frac{q}{b_2} < \epsilon, \quad 1 \le p \le b_1 - 1, \quad 1 \le q \le b_2 - 1 \ .$$

Умножая на произведение $b_1 b_2$, получим

$$-\delta < b_2 p - b_1 q < \delta, \quad 1 \le p \le b_1 - 1, \quad 1 \le q \le b_2 - 1 \ .$$

Записываем $s-1$ неравенств такого вида, по одному на каждую компоненту $b_2, \ldots, b_s$. Насколько малым следует выбирать здесь s, будет указано ниже. Первая часть алгоритма заключительно выдает все натуральные числа p, для которых существуют натуральные $q, \ldots$, такие, что все $s-1$ неравенств выполняются.

Опишем теперь вторую часть алгоритма. В ней проверяются порожденные в первой части числа p до тех пор, пока не будет получено нужное значение.

Зафиксируем p. Все точки разрыва всех n кривых, лежащие в замкнутом интервале $[p/b_1, (p+1)/b_1]$, упорядочиваются по возрастанию. Пусть x_i и x_{i+1} — две соседние точки в отсортированном списке точек. Тогда в интервале $[x_i, x_{i+1}]$ каждая из кривых b_i представляет собой прямолинейный отрезок, записываемый уравнением $b_i u - c_i^j$, где c_i^j — константа, зависящая от i и j (и, конечно, также от p).

Решением следующей системы линейных неравенств относительно u является некоторый (возможно, пустой) открытый интервал $]x_i, x_{i+1}[$:

$$x_j \le u \le x_{j+1} \ ,$$

$$\sum_{i=1}^{n} (b_i u - c_i^j) < 1 \ ,$$

$$(b_1 u - c_1^j) + \ldots + (b_{i-1} u - c_{i-1}^j) < b_i u - c_i^j, \ i = 2, \ldots, n \ .$$

Необходимым и достаточным условием для того, чтобы два числа u и m образовывали секретную пару, будет принадлежность числа u/m построенному таким образом для некоторых p и j интервалу. Действительно, последние неравенства системы выражают условие сверхроста, а предшествующее неравенство — условие, что модуль является достаточно большим.

Таким образом, во второй части алгоритма последовательно перебираются пары (p, j), где p есть кандидат, порожденный в первой части алгоритма, а j есть индекс точки из отсортированного списка точек, соответствующего данному p. Такой поиск выполняется до тех пор, пока

будет найден непустой интервал. По крайней мере та секретная пара, которая действительно использовалась при построении криптосистемы, соответствует некоторому непустому интервалу.

Работа алгоритма во второй части равносильна отысканию рационального числа u/m из некоторого непустого интервала, о котором говорилось выше. Это задача диофантовых приближений. Первая часть, равносильная порождению заслуживающих дальнейшего исследования кандидатов p, является вариантом задачи целочисленного программирования. Обе техники для решения требуют только полиномиального времени. Напомним, что алгоритм прерывает работу, если более r кандидатов для p порождаются в первой части. В неравенствах из первой части приводилась также граница s. В [Sh2] было оценено, что если мы выберем $\delta < \sqrt{b_1/2}$, то вероятность прерывания не превосходит $(2/r)^{s-1}$. Что касается степени полинома, выражающего время работы алгоритма, то ее оценить трудно, так же как и взаимосвязь между степенью, вероятностью прерывания и значениями трех выбираемых констант δ, r и s.

С точки зрения расшифрования нет особой разницы, если вместо сверхрастущего вектора мы получим некоторую перестановку сверхрастущего вектора. Эта перестановка может быть легко найдена с помощью сортировки по возрастанию. Поскольку мы не в состоянии за полиномиальное время проанализировать все $n!$ перестановок данного вектора B, мы можем сократить число анализируемых перестановок, используя тот факт, что сверхрастущий вектор является также и возрастающим. Это достигается путем уменьшения интервала $[x_j, x_{j+1}]$ за счет включения точек пересечения между кривыми (в дополнение к точкам разрыва). Это увеличит возможное число интервалов с $O(n)$ до $O(n^2)$. Внутри каждого нового интервала имеется некоторое вертикальное упорядочивание всех кривых. Этот порядок дает единственную возможную перестановку компонент a_i при условии, что рассмотрение этого интервала приводит к успеху. Неравенства системы должны быть в таком случае видоизменены, поскольку условие сверхроста больше не требуется.

Пример 3.2. Наша первая иллюстрация алгоритма будет очень простой. Пусть $B = (7, 3, 2)$ открытый вектор. Конечно, с этим вектором очень легко справиться вручную, он даже в обратном порядке является сверхрастущим. Однако в случае этого вектора все вычисления могут быть представлены очень подробно. Это означает, что многие детали алгоритма станут более ясными.

Рассмотрим первую часть алгоритма. Имеем два двойных неравен-

ства

$$-\delta < 3p - 7q < \delta, \quad -\delta < 2p - 7r < \delta \ ,$$

где $1 \le p \le 6,\ 1 \le q \le 2,\ r = 1$. Мы ищем значение p, такое, что эти неравенства выполняются для некоторых q и r в указанных пределах. Нам еще нужно выбрать значение константы δ. Выбор $\delta = \sqrt{b_1/2} = \sqrt{7/2} = = 1.87$ был рекомендован выше. Однако этот выбор не порождает никаких значений p. Дело в том, что в маленьких примерах результаты асимптотического характера могут приводить к ошибкам. Мы собираемся осуществить проверку всех значений p во второй части алгоритма. В следующей таблице указано для каждого p наименьшее значение, такое, что наши неравенства, где $<$ заменено на $\le$, имеют решение для некоторых q и r. ($7r$ может быть, конечно, заменено на 7, поскольку $r = 1$ единственно возможное значение.)

p	1	2	3	4	5	6
δ	5	3	2	2	3	5

Как будет видно из дальнейшего, даже если бы мы выбрали $\delta = 2$, мы потеряли бы верное значение p.

Таким образом, для второй части алгоритма мы допускаем все значения p в качестве кандидатов. Это означает, что мы делим интервал (0,1) на подинтервалы

$$(0, \frac{1}{7}), (\frac{1}{7}, \frac{2}{7}), (\frac{2}{7}, \frac{1}{3}), (\frac{1}{3}, \frac{3}{7}), (\frac{3}{7}, \frac{1}{2}), (\frac{1}{2}, \frac{4}{7}), (\frac{4}{7}, \frac{2}{3}), (\frac{2}{3}, \frac{5}{7}), (\frac{5}{7}, \frac{6}{7}), (\frac{6}{7}, 1),$$

такие, что все три кривые b_i будут прямолинейными отрезками $b_i u - c_i^j$ в каждом подинтервале. (Как и раньше, индекс j обозначает интервал.) Мы рассматриваем здесь открытые, а не замкнутые интервалы, поскольку никакая точка разрыва кривой b_i не дает секретную пару.

Во второй части алгоритма для каждого подинтервала рассматриваются такие неравенства

$$(7u - i') + (3u - i'') + (2u - i''') < 1 \ ,$$

$$7u - i' < 3u - i' \ ,$$

$$(7u - i') + (3u - i'') < 2u - i''' \ ,$$

где константы меняются в пределах $0 \le i' \le 6,\ 0 \le i'' \le 2,\ 0 \le i''' \le 1$, в зависимости от подинтервала. Неравенства могут быть переписаны в виде

$$\begin{aligned} 12u &< i \ , \\ 4u &< j \ , \\ 8u &< k \ , \end{aligned}$$

где новые константы получены из старых: $i = 1 + i' + i'' + i'''$, $j = i' - i''$, $k = i' + i'' - i'''$. В следующей таблице перечислены значения констант для различных интервалов и указано для каждого из них и для каждого из наших трех неравенств, выполняются ли они на всем интервале (SAT), на части интервала (PART) или не выполняются ни в одной точке интервала (NOT). Интервалы приводятся указанием своего правого конца.

Интервал	1/7	2/7	1/3	3/7	1/2	4/7	2/3	5/7	6/7	1
i'	0	1	2	2	3	3	4	4	5	6
i''	0	0	0	1	1	1	1	2	2	2
i'''	0	0	0	0	0	1	1	1	1	1
i	1	2	3	4	5	6	7	8	9	10
j	0	1	2	1	2	2	3	2	3	4
k	0	1	2	3	4	3	4	5	6	7
$12u < i$	PART	PART	NOT	NOT	NOT	NOT	PART	NOT	PART	NOT
$4u < j$	NOT	PART	SAT	NOT	SAT	NOT	SAT	NOT	PART	SAT
$8u < k$	NOT	NOT	NOT	PART	SAT	NOT	NOT	NOT	PART	PART

Ясно, что NOT появляется тогда и только тогда, когда неравенство не выполняется на левом конце интервала. Аналогично SAT появляется, если и только если неравенство справедливо для правого конца интервала. Интервал I порождает секретную пару в том и только том случае, когда SAT или PART стоят в столбце для всех трех неравенств. В этом случае интересующий нас интервал есть подынтервал I.

В нашем примере единственный такой интервал начинается с 5/7. Правым концом будет точка 3/4. В нашем случае оказалось, что все неравенства привели к одному и тому же правому концу, что, вообще говоря, может и не иметь места в общем случае. Выбирая числа $8/11, 41/56, 61/84$ и $223/308$ из найденного интервала, мы получим сверхрастущие векторы

$$(1, 2, 5),\ (7, 11, 26),\ (7, 15, 38) \text{ и } (21, 53, 138)$$

соответственно. Модуль 11 является наименьшим из возможных, поскольку в интервале $(5/7, 3/4)$ нет рационального числа со знаменателем ≤ 10.

Наш второй пример связан с вектором

$$B = (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523)\ ,$$

уже упомянутым в примере 2.1. Сейчас нет никакого смысла записывать полный список точек разрыва в каждом интервале $(p/43, (p+1)/43)$.

Например, только кривая для 1523 имеет 35 точек разрыва в каждом интервале. Однако вектор B обладает достаточной криптографической слабостью, чтобы можно было сделать различные упрощения.

Неравенства первой части алгоритма могут быть записаны в следующем виде:

$$|129p - 43q| \le \delta, \quad |215p - 43r| \le \delta, \quad |473p - 43s| \le \delta \ .$$

Поскольку числа 129, 215 и 473 оказались кратными 43, имеем $p = 1$ в качестве кандидата, даже если выбирается $\delta = 0$. Мы не будем исследовать других кандидатов, и, таким образом, нас интересует только интервал $(1/43, 2/43)$. Рассмотрим точки разрыва других кривых, лежащие в этом интервале. Ближайшая к левому концу этого интервала есть точка $36/1523$. Конечно, это не обязательно, что ближайшая точка получена при использовании самого большого числа в b_i, но для данного B это так.

Теперь наш интервал — это $(1/43, 36/1523)$. В этом интервале b_i-кривые имеют вид

$$43u - 1, \quad 129u - 3, \quad 215u - 5, \quad 473u - 11, \quad 903u - 21 \ ,$$

$$302u - 7, \quad 561u - 13, \quad 1165u - 27, \quad 697u - 16, \quad 1523u - 35 \ .$$

Неравенства, выражающие величину модуля, имеют вид

$$6011u - 139 < 1 \text{ или } u < 140/6011 \ .$$

Поскольку $140/6011 < 36/1523$, мы приходим к новому интервалу $(1/43, 140/6011)$.

Приведем теперь неравенства, выражающие условие сверхроста. В левой колонке приведено неравенство, а в правой — его решение.

$$\begin{array}{lll}
129u - 3 & < 43u - 1, & u < 1/43 \ , \\
215u - 5 & < 172u - 4, & u < 1/43 \ , \\
473u - 11 & < 387u - 9, & u < 1/43 \ , \\
903u - 21 & < 860u - 20, & u < 1/43 \ , \\
302u - 7 & < 1763u - 41, & u < 34/1461 \ , \\
561u - 13 & < 2065u - 48, & u < 35/1504 \ , \\
1165u - 27 & < 262u - 61, & u < 34/1461 \ , \\
697u - 16 & < 3791u - 88, & u < 72/3094 \ , \\
1523u - 35 & < 4488u - 104, & u < 69/2965 \ .
\end{array}$$

Первые четыре неравенства выполняются на всем интервале, тогда как оставшиеся пять сужают правый конец интервала. Наименьшим среди верхних границ, полученных для u, будет

$$72/3094 = 36/1547 \ .$$

Таким образом, получен интервал $(1/43, 36/1547)$. Выбрав число $37/1590$ из этого интервала, мы получим сверхрастущий вектор

$$(1, 3, 5, 11, 21, 79, 157, 315, 664, 1331) \ .$$

Читатель может вычислить сверхрастущий вектор самостоятельно, выбрав число $720/30949$ из нашего интервала.

Наша следующая иллюстрация связана с открытым вектором

$$\begin{aligned} B \quad = \quad & (4579, 50316, 24924, 30908, 27110, \\ & 17953, 32732, 16553, 22075, 53620) \ , \end{aligned}$$

рассмотренным в примере 3.1. Этот вектор гораздо "хитрее", чем вектор B из примера 2.1, рассмотренный выше. Не вдаваясь в подробности первой части алгоритма, укажем, что в качестве возможного варианта порождается значение $p = 88$. Это приводит к интервалу $(88/4579, 89/4579)$. Три самые левые точки разрыва наших кривых в возрастающем порядке это

$$594/30908, \quad 479/24924 \quad \text{и} \quad 967/50316 \ .$$

В интервале $(88/4579, 594/30908)$ кривые имеют вид

$$4579u - 88, \quad 50316u - 966, \quad 24924u - 478 \ ,$$

$$30908u - 593, \quad 27110u - 521, \quad 17953u - 345, \quad 32732u - 629 \ ,$$

$$16553u - 318, \quad 22075u - 424, \quad 53620u - 1030 \ .$$

Сумма этих выражений должна быть меньше 1. Это дает неравенство

$$280770u < 5393 \ ,$$

которое не выполняется ни для какого u из интервала.

Рассмотрим следущие подинтервалы:

$$(594/30908, \ 479/24924) \text{ и } (479/24924, \ 967/50316) \ .$$

Правая часть вышеприведенного неравенства лежит в подинтервалах 5394 и 5395 соответственно. (Это следует из того, что константа в кривых для 30908 и 24924 возрастает на 1.) Но тем не менее неравенство не выполняется ни для какого u из подинтервала.

Продолжим изучение, взяв интервал

$$(967/50316, \ 1031/53620) \ ,$$

правый конец которого есть следущая точка разрыва. В этом интервале неравенство, выражающее ограничение на модуль, принимает вид

$$280770u < 5396 \text{ или } u < 2698/140385 \ .$$

Это приводит к новому интервалу

$$(967/50316,\ 2698/140385) \ .$$

Запишем теперь неравенства, выражающие условие сверхроста. Как и ранее, в левой колонке записано неравенство, а в правой — решение.

$$\begin{array}{lll}
50316u - 967 & < 4579u - 88, & u < 879/45737 \ , \\
24924u - 479 & < 54895u - 1055, & u < 576/29971 \ , \\
30908u - 594 & < 79819u - 1534, & u < 940/48911 \ , \\
27110u - 521 & < 110727u - 2128, & u < 1607/83617 \ , \\
17953u - 345 & < 137837u - 2649, & u < 2304/119884 \ , \\
32732u - 629 & < 155790u - 2994, & u < 2365/123058 \ , \\
16553u - 318 & < 188522u - 3623, & u < 3305/171969 \ , \\
22075u - 424 & < 205075u - 3941, & u < 3517/183000 \ , \\
53620u - 1030 & < 227150u - 4365, & u < 3335/173530 \ .
\end{array}$$

Только первое неравенство влияет на конечную точку нашего интервала. Таким образом, окончательный подинтервал будет

$$(879/45737, 2698/140385) \ .$$

Этот интервал очень мал: его граничные точки различаются на 1 только в девятом знаке. Число 1061/55207, соответствующее секретной паре криптосистемы, лежит в этом интервале. Интересно также отметить, что ни одна из граничных точек полученного интервала не является точкой разрыва и что левый конец лежит довольно далеко от исходного левого конца 88/4579.

В нашей заключительной иллюстрации будем иметь дело со вторым вектором B из примера 3.1. Не вдаваясь в подробности, отметим, что получается следующий интервал:

$$\Big(\frac{410860730108917982154}{1264891196933908912166}, \frac{410860730109349502042}{1264891196933908912166}\Big).$$

Исходная дробь u/m лежит в этом интервале, а также

$$\frac{u'}{m'} = \frac{410860730109000000000}{1264891196933908912166} \ .$$

Сокращая дробь u'/m', мы получим сверхрастущий вектор A:

$$\begin{aligned}
a'_1 &= 450448325606142\\
a'_2 &= 454908210018084\\
a'_3 &= 918736188860052\\
a'_4 &= 1837472377720104\\
a'_5 &= 3670484871028266\\
a'_6 &= 26182899405826276\\
a'_7 &= 71194348822186470\\
a'_8 &= 142388697644372940\\
a'_9 &= 303619324952515624\\
a'_{10} &= 607234190020619306\\
a'_{11} &= 1233314769589420298\\
a'_{12} &= 2466629539178840596\\
a'_{13} &= 4933254618473269250\\
a'_{14} &= 9866513696830950442\\
a'_{15} &= 19751855943672434802\\
a'_{16} &= 39522549357124227406\\
a'_{17} &= 79045103174132866784\\
a'_{18} &= 158109039358160679368\\
a'_{19} &= 316218087636090182620\\
a'_{20} &= 632436175272180365240
\end{aligned}$$

□

3.3. Теория достижимости

Как узнать, получен ли данный рюкзачный вектор B из некоторого сверхрастущего вектора сильным модульным умножением или, возможно, последовательным применением сильного модульного умножения? И если ответ на этот вопрос положителен, как найти этот сверхрастущий вектор, а также используемые модуль и множитель. Поставленные вопросы и исследуются в этом параграфе.

Постановка проблемы будет достаточно общей. Мы не будем ограничивать величины компонент рюкзачного вектора относительно n. Алгоритм будет детерминированным, а оценка сложности зависит от того, как будет определен размер входных данных. Здесь следует отметить, что поставленные выше вопросы заметно отличаются от самой задачи о рюкзаке. Например, эти вопросы не становятся легче, если число компонент вектора B ограничено некоторой константой k. Действительно, для них не достаточно сделать перебор 2^k вариантов. Вообще говоря,

если ответы на эти вопросы найдены, то соответствующая задача о рюкзаке будет легкорешаемой.

По определению рюкзачный вектор B называется *супердостижимым*, если и только если существует сверхрастущий вектор A, такой, что B получается из A сильным модульным умножением. Для $r \leq 1$ будем называть вектор B *r-гипердостижимым*, если найдется последовательность векторов $A_0, A_1, \ldots, A_r = B$ такая, что A_0 сверхрастущий вектор и для всех $i = 0, \ldots, r-1$ вектор A_{i+1} получается из A_i сильным модульным умножением.

Ясно, что понятия супердостижимости и 1-гипердостижимости совпадают. Вообще говоря, вектор может быть построен так, что он будет r-гипердостижимым, $r > 1$, но в то же время он может быть еще и супердостижимым. Например, в основополагающей статье [МеН] о криптосистемах, основанных на задаче о рюкзаке, приведен пример вектора $B = (25, 87, 33)$, получающегося из сверхрастущего вектора $A = (5, 10, 20)$ с помощью двукратного сильного модульного умножения относительно пар модуль-множитель $(47, 17)$ и $(89, 3)$, причем вектор B не может быть получен из A одним сильным модульным умножением. Однако вектор B является супердостижимым, так как его можно получить из вектора $(2, 3, 66)$ сильным модульным умножением относительно пары $(99, 62)$.

Мы требуем выполнения условия сильного модульного умножения, поскольку при таких условиях применима лемма 3.1. Если мы имеем только модульное умножение, то нет гарантии, что решение задачи о рюкзаке со входом (B, β) будет совпадать с единственным решением задачи (A, α), где α получается из β соответствующим обратным модульным умножением. Такое заключение можно сделать, если исходное модульное умножение является сильным, даже при условии, что их несколько.

Следующий результат дает основное правило для построения примеров векторов, не являющихся r-гипердостижимыми.

Теорема 3.1 *Всякий r-гипердостижимый вектор является инъективным. Таким образом, любой супердостижимый вектор инъективен.*

Доказательство. Теорема является следствием следующих двух фактов (i) и (ii):

(i) Любой сверхрастущий вектор является инъективным. Действительно, алгоритм, описанный в примере 2.1, показывает, что задача о рюкзаке (A, α), где A сверхрастущий вектор, обладает самое большее одним решением.

(ii) Сильное модульное умножение сохраняет свойство инъективности. Предположим, что вектор B получается из A сильным модульным умножением относительно пары (m,t). Предположим далее, что $BC = BC'$ для некоторых двоичных векторов C и C'. Ясно, что вектор A может быть получен из B модульным умножением (m,u), где u — обратный элемент к t. Поскольку, по предположению, $uBC = uBC'$, то имеем также $AC \equiv AC' \pmod m$. Так как m превосходит сумму компонент вектора A, то последнее сравнение по модулю m должно быть просто равенством $AC = AC'$. Из (i) заключаем, что $C = C'$ и, таким образом, вектор B инъективен.

□

К примеру, если какая-то компонента вектора равна сумме некоторых других компонент, то такой вектор заведомо не может быть r-гипердостижимым.

Рассмотрим рюкзачный вектор $A = (a_1, \ldots, a_n)$, целое число $m > \max A$ и натуральное $t < m$ такое, что $(t,m) = 1$. *Растущей последовательностью ассоциированной* с тройкой (A,t,m) будем называть тройки $(A(k), t, m+kt)$, $k = 0,1,2,\ldots$, где

$$A(k) = (a_1 + k\cdot[ta_1/m], \ldots, a_n + k\cdot[ta_n/m]) \ .$$

Таким образом, растущая последовательность начинается с (A,t,m). Термины *множитель* и *модуль* относятся также и к числам t и $m+kt$ в тройке $(A(k), t, m+kt)$.

Например, если $A = (1,2,3)$, $t = 4$, $m = 5$, то растущая последовательность начинается с троек

$$((1,2,3),4,5), \quad ((1,3,5),4,9) \quad \text{и} \quad ((1,4,7),4,13) \ .$$

Если $A = (1,4,7)$, $t = 3$, $m = 8$, то растущей последовательностью будет

$$((1,4+k,7+2k),3,8+3k), \quad k = 0,1,2,\ldots \ .$$

Число i, $2 \le i \le n$, будем называть *точкой нарушения* в рюкзачном векторе A, если

$$a_i \le \sum_{j=1}^{i-1} a_j \ .$$

Таким образом, i-я компонента A нарушает условие сверхроста для A. Если A возрастающий вектор, то точка нарушения i в A удовлетворяет неравенству $i \ge 3$.

Целью тройки (A, t, m) будем называть первую тройку $(A(k), t, m+kt)$ в возрастающей последовательности, в которой $A(k)$ будет сверхрастущим, а $m + kt$ больше, чем сумма компонент $A(k)$, при условии, что такие тройки существуют. Ясно, что некоторые тройки сами могут быть своими целями, а для других цели вообще могут не существовать. В частности, если вектор A не является возрастающим, то (A, t, m) не обладает целью. Это следует из того, что неравенство $a_i > a_{i+1}$ влечет неравенство $[ta_i/m] \geq [ta_{i+1}/m]$ и, следовательно, для всех k

$$a_i + k \cdot [ta_i/m] > a_{i+1} + k \cdot [ta_{i+1}/m] \ .$$

Возвращаясь к двум примерам, рассмотренным выше, $i = 3$ есть точка нарушения в начальном векторе первой последовательности. Третья тройка есть цель данной последовательности. Вторая последовательность не обладает целью, поскольку модуль никогда не станет достаточно большим.

Далее мы определим понятие в некотором смысле двойственное к понятию возрастающей последовательности. Пусть (A, t, m) есть тройка, определяемая в связи с возрастающей последовательностью. *Убывающей последовательностью*, согласованной с тройкой (A, t, m), будем называть последовательность троек $(A(-k), t, m - kt)$, $k = 0, 1, 2, \ldots$, где векторы $A(-k)$ определяются с помощью (убывающей) индукции следующим образом. $A(-0) = A$. Пусть $A(-k) = (d_1, \ldots, d_n)$ уже определена и все еще выполняется $m - kt > \max A(-k)$. (Это неравенство выполняется для $k = 0$ по выбору исходной тройки.) Тогда

$$A(-k-1) = (d_1 - [td_1/(m-kt)], \ldots, dz - n - [td_n/(m-kt)]) \ .$$

Убывающие последовательности будут всегда конечными, в то время как возрастающие последовательности являются бесконечными. Однако в дальнейшем только конечные начальные сегменты возрастающих последовательностей будут представлять интерес. Теперь перейдем к разработке некоторой техники, необходимой для наших алгоритмов. Начнем со свойств возрастающих последовательностей. В леммах 3.2–3.4 обозначения $A, t, m, A(k)$ те же, что и в приведенных определениях.

Лемма 3.2 *Если вектор A является возрастающим или сверхрастущим, то каждый вектор в возрастающей последовательности, согласованной с* (A, t, m)*, также будет соответственно возрастающим или сверхрастущим.*

Доказательство. Неравенство $a_{i-1} < a_i$ влечет неравенство $[ta_{i-1}/m] \leq [ta_i/m]$. Так что, если A — возрастающий вектор, таким будет и каждый вектор $A(k)$.

Предположим теперь, что

$$\sum_{j=1}^{i-1} a_j < a_i.$$

Следовательно,

$$\sum_{j=1}^{i-1}[ta_j/m] \leq \Big[\frac{\Big(t\sum_{j=1}^{i-1} a_j\Big)}{m}\Big] \leq [ta_i/m] \ .$$

Это означает, что как только A будет сверхрастущим вектором, то таким будет и каждый вектор $A(k)$.

□

Лемма 3.3 *Если вектор $B = (b_1, \ldots, b_n)$ получается из A модульным умножением относительно (m, t), то B получается также из каждого $A(k)$ модульным умножением относительно $(m + kt, t)$. Это также верно и в случае, если "модульное умножение" заменить термином "сильное модульное умножение".*

Доказательство. Имеем по условию

$$b_i = (ta_i, \mathrm{mod} m), \text{ для } 1 \leq i \leq n \ .$$

Ясно,что $(t, m + kt) = 1$. Для всех k имеем

$$\begin{aligned} t(a_i + k \cdot [ta_i/m]) &= b_i + [ta_i/m] \cdot m + [ta_i/m] \cdot kt \\ &= b_i + [ta_i/m](m + kt) \ . \end{aligned}$$

Поскольку $b_i < m + kt$, получаем, что

$$(t(a_i + k \cdot [ta_i/m]), \mathrm{mod}(m + kt)) = b_i \ .$$

Это означает,что B получается из $A(k)$ модульным умножением относительно $(m + kt, t)$.

Предположим, что B получается из A сильным модульным умножением относительно (m, t). Это означает, что

$$\sum_{i=1}^{n} a_i < m \ .$$

Следовательно,

$$\sum_{i=1}^{n}(a_i + k[ta_i/m]) < m + \sum_{i=1}^{n} k[ta_i/m] \leq$$

$$\leq m + k[t(a_1 + \ldots + a_n)/m] \leq m + k \cdot [t] = m + kt \ .$$

Это означает, что B получается из $A(k)$ сильным модульным умножением относительно модуля $m + kt6$ и сомножителя t.

□

Как непосредственное следствие лемм 3.2 и 3.3 отметим, что любой сверхрастущий вектор может быть получен из *бесконечного числа* сверхрастущих векторов сильным модульным умножением. (Особым является случай, где $[ta_n/m] = 0$, и, следовательно, всегда $A(k) = A$, легко может быть рассмотрен отдельно.)

Теперь исследуем вопрос, когда тройка (A, t, m) обладает целью. Напомним, что для всякой точки нарушения i в векторе A выполняется

$$(*) \qquad a_i \leq a_1 + \ldots + a_{i-1} \ .$$

Предположим также, что

$$(*)' \qquad [ta_1/m] + \ldots + [ta_{i-1}/m] < [ta_i/m] \ .$$

Заметим,что условия (*) и $(*)'$ вовсе не противоречат друг другу, даже если бы мы имели в (*) строгое неравенство. Наименьшее целое x, такое, что

$$\sum_{j=1}^{i-1} a_j + x \sum_{j=1}^{i-1} [ta_j/m] < a_i + x[ta_i/m] \ ,$$

будем называть *спасителем* i. Более точно,

$$x = \Big[\frac{\Big(\Big(\sum_{j=1}^{i-1} a_j\Big) - a_i\Big)}{\Big([ta_i/m] - \sum_{j=1}^{i-1}[ta_j/m]\Big)} \Big] + 1 \ .$$

По (*) и $(*)'$ x — натуральное число.

Рассмотрим далее ситуацию, когда модуль недостаточно большой, т. е.

$$(**) \qquad m \leq \sum_{i=1}^{n} a_i \ .$$

Предположим также, что

$$(**)' \qquad \sum_{i=1}^{n}[ta_i/m] < t \ .$$

Тогда наименьшее целое y, такое, что

$$\sum_{i=1}^{n} a_i + \sum_{i=1}^{n}[ta_i/m] < m + yt \ ,$$

называется *спасителем* m. Точная формула для y может быть легко приведена по аналогии с формулой для x, написанной выше.

Если $(*)'$ выполняется для каждой точки нарушения i в A, тогда *спасителем* A назовем наибольшее значение спасителей для всех точек нарушения i.

Важно отметить, что если i спасается с помощью k, т. е. i не является точкой нарушения в $A(k)$, то тогда i не будет точкой нарушения и ни в каком векторе $A(k)$ при $k > k'$. Таким образом, если мы спасли несколько чисел (возможно, включая m), то мы можем идти дальше в возрастающей последовательности, пока все они не будут спасены (если это возможно). Для полноты отметим, что 0 является спасителем i (соответственно m), если условие (*) (соответственно (**)) не выполняется.

Лемма 3.4 *Тройка (A, t, m) обладает целью тогда и только тогда, когда $(*)'$ выполняется, если только (*) выполняется, и, более того, $(**)'$ выполняется в случае, если только (**) выполняется. Если эти условия выполняются, то целью будет тройка $(A(k), t, m + k_0 t)$, где k_0 есть максимум из спасителей A и m.*

Доказательство. Если k_0 определено, как в утверждении леммы, то $A(k_0)$ — сверхрастущий вектор (поскольку в нем нет точек нарушения) и $m + k_0 t$ больше суммы компонент $A(k_0)$. Определение k_0 гарантирует, что мы получим наименьшее число, удовлетворяющее этим условиям. С другой стороны, если некоторое i удовлетворяет (*), но в условии $(*)'$ мы имеем $\geq$ вместо $<$, тогда i будет точкой нарушения в каждом $A(k)$. Аналогично, если (**) выполняется, а $(**)'$ нет, то для всех k

$$\sum_{i=1}^{n}(a_i + k[ta_i/m]) \geq m + kt \ .$$

Таким образом, модуль будет слишком мал в любой тройке возрастающей последовательности.

□

Приведем теперь несколько примеров. В представленной ниже таблице заданы A, t, m, B и цель. Здесь B получается из A модульным умножением относительно (m, t). Цель всегда указывает тройку, показывающую, что B — сверхрастущий вектор. Если цель не существует, мы используем обозначение $\mathrm{NR}(i = i')$ или $\mathrm{NR}(m)$ в случае, если нет спасителя для точки нарушения i или для модуля m, т.е. $(*)'$ или $(**)'$ не выполняется. В некоторых случаях это может произойти несколько раз.

Пример 3.3.

A	t	m	B	Цель
(1,2,3)	4	5	(4,3,2)	$k = 2, (1, 4, 7), 4, 13$
(1,4,7)	3	8	(3,4,5)	$\mathrm{NR}(m)$:$0 + 1 + 2 \geq 3$
(1,5,6)	7	8	(7,3,2)	$k = 1$ спаситель $i = 3$,$\mathrm{NR}(m)$
(1,3,5)	4	9	(4,3,2)	$k = 1, (1, 4, 7), 4, 13$
(1,3,6)	3	7	(3,2,4)	$\mathrm{NR}(m)$
(2,3,4)	5	6	(4,3,2)	$\mathrm{NR}(i = 3)$, $\mathrm{NR}(m)$
(1,2,3)	5	6	(5,4,3)	$k = 1, (1, 3, 5), 5, 11$
(1,5,12)	8	13	(8,1,5)	$\mathrm{NR}(m)$
(1,2,10)	8	15	(8,1,5)	собств.цель
(1,8,13,36,57)	87	200	(87,96,131,132,159)	$k = 2, (1, 14, 23, 66, 105), 87, 374$
(1,34,67)	97	100	(97,98,99)	$k = 3, (1, 130, 259), 97, 391$
(1,15,29,44)	93	100	(93,95,97,92)	$k = 2, (1, 41, 81, 124), 93, 286$
(2,3,5,8)	4	9	(8,3,2,5)	$\mathrm{NR}(i = 4)$, $\mathrm{NR}(m)$ $k = 1$ спаситель $i = 3$

□

В первой из оставшихся трех лемм мы имеем дело со взаимодействием между множителем и модулем. Затем мы обсудим свойства убывающих последовательностей. В конце возрастающие и убывающие последовательности связываются вместе. Будем говорить, что B есть (A, t, m)-*супердостижимый* вектор, если A — сверхрастущий и B получается из A сильным модульным умножением относительно модуля m и множителя t.

Рассмотрим тройку (A, t, m), где $A = (a_1, \ldots, a_n)$ — рюкзачный вектор, $m > \max A$, $t < m$ и $(t, m) = 1$. Тройку (A_1, t_1, m_1), где

$$m_1 = t, \quad t_1 = (-m, \mathrm{mod} t) \ ,$$

$$A_1 = ([ta_1/m], \ldots, [ta_n/m]) \ ,$$

будем называть *транспонированной версией* (A, t, m).

Лемма 3.5 *Пусть* (A_1, t_1, m_1) — *транспонированная версия тройки* (A, t, m). *Если* B *получается из* A *модульным умножением (соответственно сильным модульным умножением) относительно* (m, t) *и* $\max B < t$, *тогда* B *получается также из* A_1 *модульным умножением (соответственно сильным модульным умножением) относительно* (m_1, t_1). *Если* B — *сверхрастущий вектор, то* B *является* (A', t', m')-*сверхрастущим с* $t' \leq \max B$.

Доказательство. Ясно, что $t_1 < t$. Повторяя процедуру замены тройки ее транспонированной версией, можно добраться до тройки, у которой $t' \leq \max B$.

Действительно, пусть B получается из A модульным умножением относительно (m, t) и $t > \max B$. Имеем $(ta_i, \bmod m) = b_i$ для $1 \leq i \leq n$. Это означает, что

$$t_1[ta_i/m] \equiv b_i - ta_i \equiv b_i \pmod{t} \ .$$

Поскольку $b_i \leq \max B < t$, мы можем переписать это равенство как

$$(t_1[ta_i/m], \bmod t) = b_i \ ,$$

которое показывает, что B получается из A модульным умножением относительно (m_1, t_1). Это же справедливо относительно сильного модульного умножения, поскольку если

$$m > \sum_{i=1}^{n} a_i \ ,$$

то

$$t > \sum_{i=1}^{n} ta_i/m \leq \sum_{i=1}^{n} [ta_i/m] \ .$$

Для доказательства последнего утверждения леммы 3.5, достаточно показать, что из того, что вектор A является сверхрастущим следует, что сверхрастущим будет и вектор A_1. Предположение, что A является сверхрастущим вектором, означает, что для $2 \leq i \leq n$

$$\sum_{j=1}^{i-1} ta_j/m < ta_i/m \ .$$

Отсюда

$$(*) \qquad \sum_{j=1}^{i-1} [ta_j/m] \leq [ta_i/m] \ .$$

Предположим, что в (*) имеет место равенство. Тогда

$$\sum_{j=1}^{i-1} m[ta_j/m] = m[ta_i/m]$$

и, следовательно,

$$\sum_{j=1}^{i-1}(ta_j - b_j) = ta_i - b_i \ ,$$

что может быть переписано в виде

$$b_i - \sum_{j=1}^{i-1} b_j = t\Big(a_i - \sum_{j=1}^{i-1} a_j\Big) \ .$$

Поскольку коэффициент при t положителен, отсюда получаем

$$t \le b_i - \sum_{j=1}^{i-1} b_j < b_i \le \max B \ .$$

Поскольку это противоречит предположению $t > \max B$, мы, следовательно, должны иметь строгое неравенство в (*). Так как i было произвольным, получаем, что вектор A_1 является также сверхрастущим.

□

В качестве примера возьмем вектор $B = (46, 45, 40, 30)$, который является $((4, 5, 10, 20), 49, 50)$-супердостижимым. По лемме 3.5 он также будет супердостижимым из каждой тройки

$$((3, 4, 9, 19), 48, 49), \quad ((2, 3, 8, 18), 47, 48) \quad \text{и} \quad ((1, 2, 7, 17), 46, 47) \ .$$

В последней тройке сомножитель уже $\le \max B$.

Перейдем теперь к обсуждению убывающих последовательностей.

Лемма 3.6 *Пусть вектор B получается из A модульным умножением относительно (m, t) и, кроме того, $m > 2\max B$ и $t \le \max B$. Тогда B получается также из $A(-1)$ модульным умножением относительно $(m - t, t)$. Более того, если A возрастающий вектор, то таким будет и вектор $A(-1)$.*

Доказательство. Мы используем наши обычные обозначения $A = A(0) = (a_1, \ldots, a_n)$ и $B = (b_1, \ldots, b_n)$. Тогда i-я компонента вектора $A(-1)$, $1 \le i \le n$, есть

$$a_i - [ta_i/m] \ .$$

Умножая на t и используя наше предположения, получим

$$ta_i - t[ta_i/m] = b_i + m[ta_i/m] - t[ta_i/m]$$

$$= b_i + (m-t)[ta_i/m] \equiv b_i \pmod{(m-t)} \ .$$

Так как, по предположению, $m - t > \max B \ge b_i$, то мы получаем

$$(t(a_i - [ta_i/m]), \mathrm{mod}(m-t)) = b_i \ .$$

Заметим, что

$$(*) \qquad m > 2t \text{ откуда } m - t > t \ ,$$

и ясно, что $(t, m-t) = 1$. Первое утверждение леммы теперь будет следовать отсюда, если мы покажем, что новый модуль достаточно большой. Предположим противное: $a_i - [ta_i/m] \ge m - t$ для некоторого i. Умножая это на t, используя выражение для ta_i и предположение $m > 2 \max B$, получим

$$t(m-t) \le b_i + (m-t)[ta_i/m] < \frac{m}{2} + (m-t)[ta_i/m] \ ,$$

из которого

$$\frac{m}{2} > (m-t)(t - [ta_i/m] \ ,$$

противоречащее (*), так как $t > [ta_i/m]$. Для доказательства второго утверждения леммы, обозначим $A(-1) = (e_1, \ldots, e_n)$. Пусть i — произвольное, $1 \le i \le n-1$. Поскольку $A(-0)$ является возрастающим, то

$$a_{i+1} = a_i + \alpha \text{ для некоторого } \alpha \ge 1 \ .$$

Предположим сначала, что $\alpha > 1$. Тогда

$$\begin{aligned} e_{i+1} &= a_i + \alpha - [t(a_i + \alpha)/m] \\ &\ge a_i + \alpha - (1 + [ta_i/m] + [t\alpha/m]) \\ &= e_i + (\alpha - 1) - [t\alpha/m] > e_i \ . \end{aligned}$$

Здесь первое неравенство верно, поскольку всегда $[x+y] \le [x] + [y] + 1$, а второе — потому что по (*)

$$[t\alpha/m] \le t\alpha/m < \frac{\alpha}{2} \ .$$

Предположим теперь, что $\alpha = 1$. В этом случае $[t\alpha/m] = 0$. Если

$$[t(a_i + 1)/m] = [ta_i/m] \ ,$$

мы получим сразу $e_{i+1} > e_i$. Поэтому предположим, что

$$(**) \qquad [t(a_i + 1)/m] = [ta_i/m] + 1 \ .$$

Ясно, что других случаев не осталось. Условия (**) означало бы, что $e_{i+1} = e_i$. Обозначим правую часть (**) через $\beta + 1$. Имеем

$$m\beta \leq ta_i < m(\beta + 1) \leq t(a_i + 1) \ .$$

Предположим, что $ta_i < m(\beta + \frac{1}{2})$. Отсюда по (*)

$$ta_i + 1 < m(\beta + \frac{1}{2}) + t = m(\beta + 1) + t - m/2 < m(\beta + 1)$$

получаем противоречие. Поэтому $ta_i \geq m(\beta + \frac{1}{2})$. Но теперь

$$b_i = ta_i - \beta m \geq m/2 \ .$$

Это дает $m \leq 2b_i \leq 2 \max B$ в противоречии с нашим предположением. Это означает, что (**) не выполняется.

□

Лемма 3.6 в дальнейшем будет применяться к тройкам убывающих последовательностей до тех пор, пока будет выполняться неравенство $m - kt > 2 \max B$. Таким образом, модуль будет вынужден стать $\leq 2 \max B$.

Важно отметить, что некоторые свойства, сохраняемые возрастающими последовательностями, не сохраняются убывающими последовательностями. Вектор A может быть сверхрастущим, хотя другие векторы в убывающей последовательности могут таковыми и не быть. Например, для

$$A = (1, 14, 23, 66, 105), \quad t = 87, \quad m = 374$$

получаем $B = (87, 96, 131, 132, 159)$ и, следовательно, $t \leq \max B$ и $m > 2 \max B$. Имеем

$$A(-1) = (1, 11, 18, 51, 81) \ ,$$

который не является сверхрастущим. Аналогично, вектор $(4, 3, 2)$ получается из $(1, 4, 7)$ сильным модульным умножением относительно

$(13, 4)$, но, переходя к первой тройке в убывающей последовательности, мы получим, что вектор $(4, 3, 2)$ уже не получается из $(1, 3, 5)$ сильным модульным умножением относительно $(9, 4)$ (хотя и получается простым модульным умножением, как и должно быть по лемме 3.6.) Такие негативные факты вполне естественны с точки зрения нашей последней леммы 3.7, и отражают то, что некоторые свойства сохраняются для части возрастающей последовательности. Эти же свойства теряются в соответствующей части убывающей последовательности. Второе утверждение в лемме 3.6 дает нам пример свойства, сохраняемого убывающей последовательностью. Это свойство при доказательстве основного результата нам не понадобится.

Лемма 3.7 *Рассмотрим A, B, m и t, удовлетворяющие предположению леммы 3.6. Рассмотрим возрастающую последовательность, согласованную с $(A(-1), t, m - t)$. Пусть (C, t, m), $C = (c_1, \ldots, c_n)$ будет первая тройка в этой последовательности. Тогда $C = A$.*

Доказательство. Так же как и в лемме 3.6, обозначим $A(-1) = = (e_1, \ldots, e_n)$. Для произвольного i, $1 \leq i \leq n$ обозначим компоненты a_i, c_i, e_i просто как a, c, e. По определению возрастающей и убывающей последовательностей имеем

$$c = e + [te/(m - t)] \text{ и } e = a - [ta/m] \ .$$

Для доказательства равенства $a = c$ (и, следовательно, леммы 3.7) достаточно показать, что

$$(*) \qquad [te/(m - t)] = [ta/m] \ .$$

Из лемм 3.3 и 3.6 мы знаем, что

$$ta \equiv tc \pmod{m} \text{ или } a \equiv c \pmod{m} \ .$$

Это влечет соотношение

$$(**) \qquad [te/(m - t)] \equiv [ta/m] \pmod{m} \ .$$

Выполнение условия (**), когда условие (*) не имеет места, возможно только в случае, если (ненулевая) абсолютная величина разности выражений в квадратных скобках кратна m. Покажем, что этого быть не может, так как оба выражения меньше чем m.

Действительно, поскольку $m - t > \max A(-1) \geq e$, то

$$[te/(m - t)] < t < m.$$

Оценим правую часть (**), обозначив $t/m = x$ и пользуясь неравенством $[y] \le y$. Имеем

$$\begin{aligned}[ta/m] &\le xa = x(e + [ta/m]) \le x(e + x(e + [ta/m])) \\ &\le x(e + x(e + x(e + [ta/m]))) \le e(x + x^2 + \ldots + x^p) + x^p[ta/m] \\ &\le e/(1 - x) + x^p[ta/m] = me/(m - t) + x^p[ta/m] \\ &< m + x^p[ta/m] \ .\end{aligned}$$

Это неравенство выполняется для произвольно большого p, что означает, что слагаемое $x^p[ta/m]$ может быть сделано произвольно малым. Следовательно, $[ta/m] < m$.

□

Лемма 3.7 может быть использована последовательно так же, как это было в случае леммы 3.6. Мы можем порождать убывающую последовательность до тех пор, пока модуль удовлетворяет неравенству $m - kt > 2 \max B$. Как только будет достигнуто значение s, такое, что $m - st \le 2 \max B$, мы можем пытаться снова увеличить модуль, рассматривая возрастающую последовательность. В таком случае лемма 3.7 показывает, что возрастающая последовательность совпадает с исходной убывающей последовательностью.

Приводимый далее основной результат теперь достаточно очевиден с точки зрения развитой техники.

Теорема 3.2 *Рюкзачный вектор B будет супердостижимым тогда и только тогда, когда для некоторых A, $t \le \max B$ и $m \le 2 \max B$ вектор B получается из A модульным умножением относительно (m, t) и тройка (A, t, m) обладает целью.*

Доказательство. В одну сторону доказательство вытекает из леммы 3.3 и определения цели. Лемма 3.4 дает простой способ определения, обладает ли данная тройка целью или нет.

Для доказательства достаточности предположим, что B — супердостижимый вектор. По лемме 3.5 B будет (A, t, m)-супердостижимым с $t \le \max B$. Если $t \le 2 \max B$, то все доказано. В противном случае образуем убывающую последовательность

$$(A(-k), t, m - kt), \quad 0 \le k \le s \ ,$$

где s — наименьшее целое, такое, что $m - st \le 2 \max B$. По лемме 3.7 тройка $A(-s), t, m - st)$ обладает целью.

□

Алгоритм, вытекающий из теоремы 3.2, может быть описан следующим образом. Для данного B выберем m, удовлетворяющее условиям $\max B < m \leq 2 \max B$ и $u < m$, где $(u, m) = 1$. Проверяем, будет ли вектор A, получающийся из B модульным умножением относительно (m, u), возрастающим и $u^{-1} = t \leq \max B$. Если нет, выбираем другую пару (u, m). При положительном ответе по лемме 3.4 проверяем, обладает ли тройка (A, t, m) целью. Если обладает, то B — супердостижимый вектор и цель также дает и сверхрастущий вектор, множитель и модуль, показывающие это. Если (A, t, m) не обладает целью, выбираем другую пару (u, m). Когда все возможные пары (u, m) будут безуспешно перебраны, алгоритм прекращает работу с заключением, что B не является супердостижимым вектором.

При выборе пар (u, m) могут быть сделаны различные упрощения. Алгоритм является детерминированным и работает для всех входов независимо ни от каких соглашений относительно размера компонент векторов. Можно распознать также и несупердостижимые векторы. Как будет ниже показано, подобные алгоритмы могут использоваться также и при решении других проблем.

Пример 3.4. Дадим несколько иллюстраций работы алгоритма. Рассмотрим вначале вектор $B = (4, 1, 6)$. В следующей таблице указаны все пары (u, m), где $m \leq 2 \max B$ и результирующий вектор A является возрастающим. Обозначения используем те же, что и в примере 3.3.

u, m	$t = u^{-1}$	A	Цель
3,11	4	(1,3,7)	$k = 1, (1, 4, 9), 4, 15$
9,11	5	(3,9,10)	NR$(i = 3)$, NR(m)
5,8	5	(4,5,6)	NR$(i = 3)$, NR(m)
2,7	4	(1,2,5)	$k = 2, (1, 4, 9), 4, 15$

Таким образом, $(4, 1, 6)$ супердостижимый вектор. Интересно отметить, что в общих случаях, приводящих к успеху, получается одна и та же цель. Из таблицы видно, что как только вектор $(4, 1, 6)$ является (A, t, m)-супердостижимым, то $t \geq 4$ и $m \geq 15$. Таким образом, не достаточно брать только модули $m \leq 2 \max B$ без рассмотрения возрастающих последовательностей. Конечно, m может быть произвольно большим в возрастающей последовательности. Также и t может быть сделано большим с использованием аргументов аналогичным тем, что были приведены в лемме 3.5, только в обратном порядке.

Вектор $B = (1, 10, 8)$ является 2-гипердостижимым, так как он получается из $(1, 2, 4)$ двумя сильными модульными умножениями, вначале относительно $(8, 5)$ и затем относительно $(12, 5)$. Следующая таблица показывает, что B не является сверхрастущим.

u,m	$t=u^{-1}$	A	Цель
7,20	3	(7,10,16)	NR($i=3$), NR(m)
9,20	9	(9,10,12)	NR($i=3$), NR(m)
2,17	9	(2,3,16)	NR(m)
6,17	3	(6,9,14)	NR($i=3$), NR(m)
5,14	3	(5,8,12)	NR($i=3$), NR(m)
3,13	9	(3,4,11)	NR(m)
4,11	3	(4,7,10)	NR($i=3$), NR(m)
5,11	9	(5,6,7)	NR($i=3$), NR(m)

Среди рюкзачных векторов с компонентами ≤ 4 только следующие являются супердостижимыми:

$$(2,4,3),\ (4,3,2),\ (1,2,4),\ (2,4,1),\ (4,1,2)\ .$$

Изучение вектора $(4,3,2)$ показывает, что нельзя отбрасывать неинъективные кандидаты A, несмотря на теорему 3.1. Это связано с тем, что инъективность может быть приобретена позже в возрастающей последовательности.

Вернемся теперь к примеру 3.2 и покажем, как некоторые результаты можно получить с помощью метода теоремы 3.2.

Рассмотрим $B=(7,3,2)$. Мы видели, что число $61/84$ лежало в построенном интервале. Так как 73 — обратный элемент к 61, заключаем, что B — $((7,15,38),73,84)$-супердостижимый. Здесь множитель слишком большой. Лемма 3.5 последовательно дает тройки

$$((6,13,33),73)\ ,\ (15,11,28),51,62)\ ,$$

$$((4,4,23),40,51)\ ,\ ((3,7,18),29,40)\ ,$$

$$((2,5,13),18,29)\ ,\ ((1,3,8),7,18)\ .$$

В последней тройке множитель $t=7$ удовлетворяет неравенству $t\leq \max B$, и мы не можем применить лемму 3.5. Однако у нас все еще $m>2\max B$. Делая один шаг в убывающей последовательности, получаем тройку $((1,2,5),7,11)$.

Рассмотрим в заключение вектор

$$B=(43,129,215,473,903,302,561,1165,697,1523)\ .$$

В примере 3.2 мы вычисляли интервал $(1/43,36/1547)$. Выбирая число $u/m=72/3092$ из этого интервала, получим сверхрастущий вектор

$$A=(1,3,5,11,21,79,157,315,664,1331)\ .$$

Здесь $t = 43 < \max B$, но $m > 2 \max B$. Делая два шага назад в убывающей последовательности, получим тройку

$$((1, 3, 5, 11, 21, 77, 153, 307, 646, 1295), 43, 3009).$$

Теперь и m также находится в нужных пределах.

□

Будем называть вектор B *перестановочно-супердостижимым*, если некоторая из его перестановок является супердостижимым вектором. Криптоаналитическое значение перестановочно-супердостижимых векторов уже обсуждалось ранее. Как и в теореме 3.1, мы можем показать, что каждый перестановочно-супердостижимый вектор будет инъективным. Наоборот, из нашей теории легко вытекает, что, например, каждый ннъективный вектор (b_1, b_2, b_3) будет перестановочно-супердостижимым.

Предположим, что вектор B является супердостижимым. Теорема 3.2 дает метод отыскания наименьшего m, такого, что B будет (A, t, m)-супердостижимым для некоторых A и t. Множитель t может быть аналогично минимизирован. Оценивая максимальное число шагов в возрастающей последовательности до достижения цели, можно вычислить также верхнюю оценку M, зависящую от B, такую, что B будет супердостижимым тогда и только тогда, когда он будет (A, t, m)-супердостижимым с $m \leq M$. Используя наши леммы, можно также решить по данной паре (B, r), будет ли вектор B r-гипердостижимым, и в случае положительного ответа построить соответствующие сверхрастущий вектор, множители и модули. Более детально обо всем этом можно посмотреть в [Sa4]. В следующем параграфе будет показано, что для получения открытого вектора можно выбрать произвольный начальный вектор, если применить к нему достаточно много раз сильное модульное умножение.

3.4. Новая попытка скрыть лазейку

В оставшихся двух параграфах этой главы обсуждаются варианты рюкзачной криптосистемы, демонстрирующие различные способы отражения криптоаналитических атак.

Как уже неоднократно подчеркивалось, в криптографии необходима определенная осторожность в использовании результатов, основанных

на теории сложности. С криптографической точки зрения нельзя проводить анализ, основываясь на том, что в "наихудших" случаях сложность проблемы велика при недостаточной информации или ее отсутствии о сложности проблемы в среднем. Что же касается полиномиальных алгоритмов, то степень полинома, оценивающего сложность алгоритма, тоже является важной информацией. И даже если предполагаемое криптоаналитическое нападение приведет к NP-полной проблеме, в принципе, могут быть другие нападения, приводящие к легким задачам. Проиллюстрируем сказанное, используя идеи, основанные на задаче о рюкзаке. Описываемая криптосистема будет открытой только частично в том смысле, что рюкзачный вектор $A = (a_1, \ldots, a_n)$ будет открытым, в то время как некий ключ $K = (k_1, \ldots, k_n)$ будет секретным, где $k_i = 0, 1$. Этот ключ используется как при зашифровании, так и при расшифровании. Условие "выбираемый открытый текст" при криптоанализе ведет, по-видимому, к NP-полной задаче, в то время как условие "известный открытый текст" с достаточно длинным открытым текстом приводит к легкой задаче.

Будем использовать знак $\oplus$ для обозначения побитового сложения по $\bmod 2$. Это обозначение распространяется также и на векторы. Так, $1 \oplus 1 = 0$ и $(1,1,0,1,0) \oplus (1,1,1,0,0) = (0,0,1,1,0)$. Обозначим

$$t = \left\lfloor \log_2 \Big(1 + \sum_{i=1}^{n} a_i\Big) \right\rfloor + 1 \ .$$

Понятно, что любая сумма a_i-х, где каждая a_i появляется не более одного раза, выражается как двоичное число с t разрядами.

Как уже упоминалось, A будет окрытым, а двоичный вектор K — секретным. Для зашифрования исходный текст делится на двоичные блоки вида $P = (p_1, \ldots, p_t)$ длины t. Для каждого P выбирается свой случайный двоичный вектор $R = (r_1, \ldots, r_n)$. Далее образуется сумма

$$A(K \oplus R) = \sum_{i=1}^{n} (k_i \oplus r_i) a_i \ .$$

(Здесь $K \oplus R$ рассматривается как вектор-столбец.) Пусть S будет двоичным представлением этой суммы длины t с добавленными слева нулями, если это необходимо. Зашифрованная версия P теперь имеет вид

$$C = (L, R), \text{ где } L = S \oplus P \ .$$

Таким образом, $(n+t)$-битовый криптотекст соответствует t-битовому исходному тексту. Поскольку последние n битов криптотекста дают K,

то законный получатель, знающий K, может немедленно вычислить S и, таким образом, исходный текст P по t-битовому началу L криптотекста.

Криптоаналитик, знающий некоторую пару (P, C), где P может даже выбираться, немедленно может вычислить S из $L \oplus P = S \oplus P \oplus P = S$. Однако полученное таким способом S соответствует одному конкретному тексту P. И хотя R известно, определение ключа K, по-прежнему, приводит к NP-полной задаче о рюкзаке. Таким образом, криптоаналитик не извлекает информации для расшифрования другого криптотекста, полученного позже.

Предположим, однако, что криптоаналитику известна некоторая пара (текст, криптотекст) с достаточно длинным исходным текстом. Более точно, она должна состоять из n t-битовых блоков. Это означает, что криптоаналитику известно n троек $(P_i, L_i, R_i), i = 1, \ldots, n$.

Обозначим покоординатное умножение двух n-битовых векторов, T и U через $T * U$. Так, i-я компонента в $T * U$ равна 1, если i-я компонента в T и U равна 1. Легко видеть, что

$$T \oplus U = T + U - 2(T * U) \ .$$

Действительно, для $n = 1$ это очевидно. Предполагая равенство для двух n-битовых векторов, докажем его для двух $n+1$-битовых векторов, применяя предположение индукции к последним n битам (результатом будет n-битовый вектор без переносов), после чего для старших битов дело сводится к случаю $n = 1$. Конечно, $+$ и $-$ обозначают выше обычное сложение и вычитание. Например,

$$11010 \oplus 10111 = 01101 = 13 = 11010 + 10111 - 2 \cdot 10010 = 26 + 23 - 2 \cdot 18 \ ,$$

где двоичные векторы записаны без скобок и запятых.

Криптоаналитик теперь может записать n линейных уравнений

$$S_i = A(K + R_i) = A(K + R_i - 2(K * R_i)), \quad 1 \le i \le n \ ,$$

для n неизвестных k_i. Если определитель этой системы не равен 0, K может быть быстро вычислено. С другой стороны, если система окажется вырожденной, то, располагая еще несколькими тройками (P_i, L_i, R_i), придем, весьма вероятно, к невырожденной системе. Действительно, если известно $n + j$ троек, то вероятность получения невырожденной системы стремится к 1 с ростом j очень быстро.

Как пример рассмотрим $A = (2, 3, 4, 5, 6, 7)$, и, значит, $n = 6$ и $t = 5$. $K = 110011$ выбран как секретный ключ. Заметим, что в этой криптосистеме инъективность вектора A уже не требуется, так как процесс

расшифрования дает те компоненты A, которые суммируются, и, таким образом, задача о рюкзаке вообще не решается.

Зашифруем исходный текст $P_i = 01010$, выбрав $R_1 = 101010$. Имеем $K \oplus R_1 = 011001$, следовательно $S_1 = 3 + 4 + 7 = 01110$ и $C_1 = 00100101010$. (Индекс 1 в S_1 указывает на взаимосвязь с R_1.) Зная P_1 и C_1 криптоаналитик может немедленно вычислить $S_1 = 00100 \oplus$ $\oplus 01010 = 01110$. Теперь следует решать задачу о рюкзаке $(A, 14)$, чтобы найти $K \oplus R_1$, из которого можно получить K, так как $K \oplus R_1 \oplus$ $R_1 =$ $= K$. Вектор R_1, конечно, берется немедленно из C_1. Поэтому знание пары (P_1, C_1) не дает в принципе возможности расшифровать криптотексты

$$C_2 = 11110010101,\ C_3 = 01110111101,\ C_4 = 00111011110,$$

$$C_5 = 11110001010,\ C_6 = 00111011011\ .$$

Предположим, однако, что криптоаналитик знает шесть пар (P_i, C_i), $1 \le i \le 6$, где

$$P_2 = 10011,\ P_3 = 00001,\ P_4 = 10101,\ P_2 = 01110,\ P_6 = 00001\ .$$

(Это не случайное совпадение, что исходный текст $P_2 - P_6$ представляет двоичное кодирование слова SAUNA.) Теперь может быть записана система 6 линейных уравнений для неизвестных k_i. Возьмем $i = 1$. Как выше, получим

$$S_1 = 14 = AR_1 + A(K - 2(K * R_1))\ ,$$

откуда

$$2 = -2k_1 + 3k_2 - 4k_3 + 5k_4 - 6k_5 + 7k_6\ ,$$

и аналогично из уравнений для $S_2 - S_6$

$$\begin{aligned}
-2 &= 2k_1 - 3k_2 + 4k_3 - 5k_4 + 6k_5 - 7k_6\ ,\\
-6 &= -2k_1 - 3k_2 - 4k_3 - 5k_4 + 6k_5 - 7k_6\ ,\\
0 &= 2k_1 - 3k_2 - 4k_3 - 5k_4 - 6k_5 + 7k_6\ ,\\
6 &= 2k_1 + 3k_2 - 4k_3 + 5k_4 - 6k_5 + 7k_6\ ,\\
-14 &= 2k_1 - 3k_2 - 4k_3 + 5k_4 - 6k_5 - 7k_6\ .
\end{aligned}$$

Эта система, очевидно, является вырожденной. Однако она дает единственное решение для k. Действительно, из третьего и пятого уравнений следует, что $k_3 = 0$, а из второго и пятого, — что $k_1 = 1$. Дальнейший анализ основан на том, что k_i являются двоичными переменными. Непосредственно видно, что только одно из неизвестных

k_2, k_4, k_6 равно 0. Последнее уравнение с подставленными вместо k_1 и k_3 значениями дает

$$-16 = 3k_2 + 5k_4 - 6k_5 - 7k_6 \ ,$$

которое показывает, что k_4 должно быть единственной переменной равной 0. Это означает, что оставшиеся переменные должны равняться 1.

В этом примере единственное двоичное решение было получено, хотя над полем рациональных чисел единственности решения нет. Рассмотренный криптоаналитический метод имеет сходство с методом, использованным в связи с криптосистемой Хилла из примера 1.2.

Далее мы обсудим понятие в некотором смысле более слабое, чем r-гипердопустимость. Открытым ключом будет рюкзачный вектор, полученный последовательным сильным умножением из некоторого вектора, необязательно сверхрастущего. Модули и сомножители составляют секретную лазейку. Этой информации достаточно легальному получателю для расшифрования с использованием системы линейных уравнений.

Опишем необходимые детали. Разработчик криптосистемы выбирает произвольный инъективный рюкзачный вектор $A_1 = (a_1^1, \ldots, a_n^1)$, сомножитель t_1 и модуль m_1, удовлетворяющие условиям сильного модульного умножения, т. е.

$$1 \le t_1 < m_1, \quad (t_1, m_1) = 1, \quad m_1 > \sum_{i=1}^{n} a_i^1.$$

Предположим, что $A_2 = (a_1^2, \ldots, a_n^2)$ получается из A_1 сильным модульным умножением относительно пары (m_1, t_1). Теперь выбираем t_2 и m_2 так, чтобы выполнялись условия сильного модульного умножения (для A_2). Пусть $A_3 = (a_1^3, \ldots, a_n^3)$ есть вектор, получающийся из A_2 сильным модульным умножением относительно (m_2, t_2). Эта процедура повторяется до тех пор, пока не будет найден вектор $A_n = (a_1^n, \ldots, a_n^n)$, получающийся сильным модульным умножением из A_{n-1} относительно (m_{n-1}, t_{n-1}). Создатель криптосистемы (который в этом случае рассматривается как и легальный получатель сообщений) открывает вектор A_n как ключ для зашифрования, а пары (m_i, t_i), $1 \le i \le n-1$ держат в секрете в качестве лазейки.

С помощью секретной лазейки можно немедленно вычислить обратные элементы u_i для $t_i \pmod{m_i}$, $1 \le i \le n-1$. Получив криптотекст α_n, легальный получатель должен отыскать n двоичных переменных $x_1, \ldots, x_n$, таких, что

$$(*) \qquad \sum_{i=1}^{n} a_i^n x_i = \alpha_n \ .$$

После $n-1$ модульных умножений определим числа α_i, удовлетворяющие

$$\alpha_i = (u_i, \alpha_{i+1}, \mathrm{mod} m_i),\ 1 \le i \le n-1\ .$$

Эти числа α_i образуют правые части уравнений, полученных из $(*)$ последовательными модульными умножениями с использованием обратных сомножителей. На самом деле будут получены только сравнения по модулю m_i, но сравнения сводятся к обычным равенствам в силу леммы 3.1. Таким образом, легальный получатель получает систему n линейных уравнений

$$\sum_{i=1}^{n} a_i^j x_i = \alpha_j,\ j = 1, \ldots, n\ .$$

Из этой системы могут быть вычислены неизвестные x_i с оговоркой относительно возможной вырожденности системы, уже упоминавшейся выше. Эта оговорка не слишком ограничивает нас, поскольку дополнительно известно, что все x_i являются двоичными переменными. Однако если исходный вектор A_1 не является инъективным, то неоднозначность будет сохраняться после применения (сильного) модульного умножения и, таким образом, будет присутствовать в каждом уравнении системы. Криптоаналитик, пытающийся применить алгоритм типа тех, что рассматривались в параграфах 3.2 и 3.3, столкнется с определенными трудностями, так как в этой ситуации нет никакого сверхрастущего вектора и искать нечего.

В качестве простого примера рассмотрим

$$\begin{array}{lll} A_1 = (3,2,6), & t_1 = 13, & m_1 = 19, \\ A_2 = (1,7,2), & t_2 = 2, & m_2 = 11, \\ A_3 = (2,3,4), & \textit{открывается}. & \end{array}$$

Имеем $u_2 = 6$ и $u_1 = 3$. Криптотекст 6 приводит к системе уравнений

$$\begin{array}{rcl} 2x_1 + 3x_2 + 4x_3 & = & 6\ , \\ x_1 + 7x_2 + 2x_3 & = & 3\ , \\ 3x_1 + 2x_2 + 6x_3 & = & 9\ , \end{array}$$

из которой получается единственный двоичный вектор 101, хотя исходная система является вырожденной и обладает общим решением $x_2 = 0, x_1 = 3 - 2x_3$.

Последующая рюкзачная система может быть использована также для электронных подписей в смысле, уже обсуждавшемся в параграфе 2.3. Две основные цели криптографии, секретность и схемы порождения электронной подписей, являются в определенном смысле

конфликтными, и, таким образом, трудно полностью достичь их обеих в рамках одной и той же криптосистемы. Большинство вариантов рюкзачных криптосистем удовлетворяют, как предполагается, требованию секретности. Следующая конструкция представляется особенно удобной для порождения подписей. Особое значение имеют ее скорость и простота. Как подписание, так и проверка могут быть выполнены, пользуясь только сложением и вычитанием.

Нам понадобится следующая модификация задачи о рюкзаке, которая, как легко можно показать, также является NP-полной.

Для $n \geq 3$ даны $n+2$ натуральных числа $a_1, \ldots, a_n, \alpha$ и m, причем все a_i различны и $m > \max\{a_i | 1 \leq i \leq n\}$. Требуется найти (если возможно) решение $(c_1, \ldots, c_n)$ сравнения

$$\sum_{i=1}^{n} a_i c_i \equiv \alpha \pmod{m} ,$$

где каждая c_i удовлетворяет условию $0 \leq c_i \leq [\log_2 m] + 1$. Таким образом, числа a_i могут участвовать в сумме несколько раз. Однако число повторений является малым и не превосходящим числа разрядов в модуле.

До того как представить формальные детали, обсудим в общих терминах, как такая рюкзачная система может использоваться для порождения подписей. Отправитель выбирает и открывает рюкзачную систему, определяемую вектором $A = (a_1, \ldots, a_n)$ и числом m таким, что возникает трудная задача о рюкзаке, хотя она на самом деле может быть решена быстро с использованием некоторой секретной информации. Используя эту секретную лазейку и решая $(*)$, посылающий подписывает сообщение x: набор $(c_1, \ldots, c_n)$ образует подпись для сообщения α. Легальный получатель, получивший как α, так и подпись, может проверить подпись, подставляя в $(*)$. Если же легальный получатель или криптоаналитик захотят подделать подпись отправителя для некоторого сообщения α', то он должен решить задачу о рюкзаке, определенную тройкой (A, m, α). Дополнительное требование к выбору рюкзачной системы состоит в том, что все планируемые сообщения α должны иметь подпись, т.е. чтобы $(*)$ имело бы решение для всех таких α.

Теперь мы готовы представить все формальные детали с позиции легального отправителя, который в этом случае является и создателем криптосистемы. Рассмотрим простое число m, двоичное представление которого имеет длину t. (Обычно, $t = 200$.) Пусть $H = (h_{ij})$ матрица размера $t \times 2t$, элементы которой случайно выбираются из $\{0,1\}$, и A

вектор-столбец размерности $2t$, удовлетворяющий следующим t сравнениям:

$$HA \equiv \begin{pmatrix} 2^0 \\ 2^1 \\ \vdots \\ 2^{t-1} \end{pmatrix} \pmod{m}.$$

Здесь только t сравнений с $2t$ неизвестными, т.е. компонентами A. В принципе t компонент A можно выбрать случайно и вычислить оставшиеся. Такое вычисление может быть выполнено быстро, а вероятность столкнуться здесь с какой-нибудь проблемой минимальна, поскольку случайно выбираемые элементы могут быть изменены, как только это понадобится. Мы выбираем $0 \le i \le t-1$ и $1 \le j \le 2t$ для обозначения строк и столбцов матрицы.

Компоненты a_i вектора A будут выглядеть как случайные числа, причем любая степень 2 от 2^0 до 2^{t-1} может быть выражена как сумма по модулю m некоторых из них.

Вектор A и число m открываются, в то время как H держится в секрете как лазейка. Сообщениями α будут числа из отрезка $[1, \ldots, m-1]$. Подписью для α будет вектор $C = (c_1, \ldots, c_{2t})$, удовлетворяющий $(*)$, где $n = 2t$.

Подпись можно немедленно проверить с помощью $(*)$. Подделка подписей по указанным выше причинам будет труднорешаемой задачей. Действительно, для этого придется решить NP-полную модульную задачу о рюкзаке.

С другой стороны, порождение подписи выполняется быстро с использованием секретной лазейки H. Для того чтобы подписать сообщение α, мы записываем α как сумму степеней 2:

$$\alpha = \sum_{t=0}^{t-1} b_i 2^i \ .$$

Таким образом, b_i есть $(i+1)$-й разряд справа в двоичном представлении α. Число разрядов t будет достаточно из-за ограничений на α. Мы утверждаем, что можно выбрать

$$c_j = \sum_{i=0}^{t-1} b_i h_{ij}, \ 1 \le j \le 2t \ .$$

Имеем, что c_j не превосходит числа разрядов t в двоичном представлении m, как требуется в $(*)$. Более того,

$$\begin{aligned}\alpha &= \sum_{i=0}^{t-1} b_i 2^i \equiv \sum_{i=0}^{t-1} b_i \sum_{j=1}^{2t} a_j h_{ij} = \\ &= \sum_{j=1}^{2t} \left(\sum_{i=0}^{t-1} b_i h_{ij} \right) a_j = \sum_{j=1}^{2t} c_j a_j \pmod{m} .\end{aligned}$$

Для порождения и проверки подписей дополнительно необходимо только выполнение сложений.

Приведенная система не предназначена для того, чтобы скрыть сообщения, поскольку они посылаются открытым текстом. Что касается безопасности процедуры подписи, то возможно нападение, основанное на линейном алгоритме. Действительно, накопив достаточно много пар сообщение-подпись, можно вычислить матрицу H. Ситуация аналогична той, что была в нашей первой системе из этого параграфа, а также в системе Хилла из примера 1.2.

Проблема с безопасностью может быть решена путем рандомизации сообщения α перед его подписанием. Это может быть сделано, например, вычитанием из α случайно выбранного подмножества a_i-х:

$$\alpha' = \left(\alpha - \sum^{2} t_{j=1} r_j a_j, \mathrm{mod} m \right) ,$$

где $R = (r_1, \ldots, r_{2t})$ — случайный двоичный вектор. Потом находится подпись C' для α' описанным выше способом. После чего $C' + R$ можно использовать как подпись для α, поскольку

$$\alpha \equiv \alpha' + RA \equiv C'A + RA = (C' + R)A \pmod{m} .$$

Компоненты новой подписи по-прежнему будут в позволяемых пределах. Отметим, что случайный вектор R не нужно знать даже легальному получателю.

Пример 3.5. Рассмотрим модуль $m = 29$ с двоичным представлением 11101. Таким образом, H будет случайной матрицей размера 5×10. Выберем

$$H = \begin{pmatrix} 1\,0\,1\,1\,0\,1\,1\,1\,0\,0 \\ 0\,1\,0\,1\,1\,1\,0\,0\,1\,0 \\ 1\,1\,1\,0\,0\,0\,0\,1\,1\,1 \\ 0\,0\,1\,1\,0\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0\,0\,1 \end{pmatrix} .$$

Возьмем, например,

$$A = (14, 15, 19, 16, 3, 24, 10, 5, 2, 7)$$

и все пять сравнений будут выполнены. Например, третье сравнение имеет вид

$$14 + 15 + 19 + 5 + 2 + 7 = 62 \equiv 4 \pmod{29} .$$

Для сообщения $\alpha = 22$, записываемого в двоичном виде в обратном порядке как 01101, используя H, немедленно получим подпись $C = 2322210122$. Правильность подписи проверяется посредством

$$CA = 196 \equiv 22 \pmod{29} .$$

Аналогично, исходные тексты 9, 8, 20 и 1 имеют в качестве подписей

$$1022021101,\ 0011010001,\ 2221100112 \text{ и } 1011011100 .$$

Исходные тексты могут рассматриваться как закодированные числами буквы. Это означает, что слово VIHTA имеет в качестве подписи найденные пять подписей, записанных одна за другой. Причем никакой путаницы не должно возникать, даже если мы не используем отметок границ.

Рассмотрим, наконец, рандомизированную подпись для $\alpha = 22$. Выберем десять случайных бит следующим образом: $R = 1011010000$. Получим $\alpha' = (22 - 73, \mathrm{mod} 29) = 7$. Подписью для α' будет 2222121221 и, следовательно, рандомизированной подписью для α будет 3233131221. Читатель может породить такие рандомизированные подписи и для остальных букв исходного текста I,H,T,A. Отметим, что один и тот же исходный текст имеет несколько рандомизированных подписей.

□

Последняя криптосистема, представленная в этом параграфе, скрывает очень простым способом тот факт, что начальный вектор A — сверхрастущий. Это выполняется добавлением случайного шума к компонентам A так, что новые компоненты уже не образуют сверхрастущей последовательности, а только некие сегменты в их двоичных представлениях образуют такие последовательности. После перемешивания в результате сильного модульного умножения эти сегменты становятся уже невидимыми.

Опишем теперь детали. Как обычно, пусть n обозначает число компонент рассматриваемых рюкзачным вектором. Обозначим $g = [\log_2 n] + 1$. Имеем $n < 2^g$. Пусть r_1 и r_2 — произвольные натуральные числа. Выберем случайно целые R_1^i и R_2^i, удовлетворяющие $0 \le R_j^i < 2^{r_j}$, $1 \le j \le 2$, $1 \le i \le n$. Определим

$$a_i = R_1^i 2^{n+g+r_2} + 2^{g+r_2+i-1} + R_2^i .$$

a_i-е, где $1 \le i \le n$, могут быть изображены следующим образом:

Номера разрядов	r_1	n	g	r_2
Разряды	R_1^i	$0\dots010\dots0$ $n\dots i\dots1$	$0\dots0$	R_2^i

Назначение R_1^i состоит в сокрытии того факта, что a_i-е являются сверхрастущими, это обстоятельство становится сразу же известным, если каждое из R_1^i равно 0. Назначение другого случайного блока R_2^i — это скрыть результаты сильного модульного умножения.

g-Блок из нулей служит буферной зоной для сложения чисел R_2^i: он предотвращает переносы внутрь n-разрядного единичного блока, который выражает условия сверхроста. Действительно,

$$\sum_{i=1}^{n} R_2^i < n2^{r_2} < 2^{g+r_2} \ .$$

Пусть t и m удовлетворяют условию сильного модульного умножения для $A = (a_1, \dots, a_n)$. Тогда получающийся из A сильным модульным умножением относительно пары (t, m) вектор $B = (b_1, \dots, b_n)$ открывается как ключ для зашифрования. Числа t, m, r_1, r_2 образуют секретную лазейку. (Число g известно, поскольку оно задается через n.) Для легального получателя, знающего лазейку, расшифрование выполняется тривиально. Пусть n — обратный к $t \bmod m$. Для криптотекста β обозначим

$$\alpha = (u\beta, \mathrm{mod} m) \ .$$

Тогда исходный текст, соответствующий β, есть просто n разрядный блок в двоичном представлении α, получаемый зачеркиванием последних $r_2 + g$ разрядов и считыванием в обратном порядке n разрядов с конца оставшейся части числа. (Мы могли бы ввести буферную зону также и для последовательности R_1^i, но на самом деле это не нужно.)

Итак, легальное расшифрование проще, чем в основной рюкзачной криптосистеме из параграфа 3.1, поскольку здесь сверхрастущий вектор состоит из степеней 2 и, следовательно, двоичный вектор суммы непосредственно дает верную последовательность битов. Здесь мы обсудили только основной вариант, в котором сверхрастущий вектор наипростейший. Общий случай произвольного сверхрастущего вектора будет громоздким, поскольку в этом случае требуется несколько разрядов для представления компонент вектора. Алгоритм типа того, что был описан в параграфе 3.2, не работает для криптосистем этого вида.

Пример 3.6. Выбираем $n = 5$, следовательно, $g = 3$. Открытый ключ шифрования — это вектор

$$B = (62199,\ 61327,\ 13976,\ 16434,\ 74879)\ .$$

Легальный получатель знает также лазейку

$$m = 75000,\ n = 22883 (t = 1547),\ r_2 = 4\ .$$

Предположим, что получен криптотекст 151054. При умножении по модулю 75000 на 22883 получается число 43682. Двоичное представление этого числа есть

$$43682 = 1010\ 10101\ 010\ 0010\ ,$$

где выделены четыре различных блока. Получатель удаляет $r_2 + g = 7$ разрядов с конца. Следующие 5 разрядов дают исходный текст 10101. Это можно и проверить: $62199 + 13976 + 74879 = 151054$.

Аналогично модульное умножение, примененное к криптотексту 75303 дает 33549 или в двоичной записи

$$33549 = 1000\ 00110\ 000\ 1101\ .$$

Исходный текст теперь 01100, который возникает после удаления 7 разрядов, если взять 5 разрядов с конца в обратном порядке. Технически это вызвано тем, что мы считываем сверхрастущую часть не в том порядке. Таким образом, сейчас 75303 есть сумма второй и третьей компонент вектора B, как это и должно быть.

Запишем еще исходный вектор A вместе с двоичными представлениями, поделенными на блоки.

	$r_1 = 3$	существен.	$g = 3$	$r_2 = 4$
$a_1 = 24717$	110	00001	000	1101
$a_2 = 20741$	101	00010	000	0101
$a_3 = 12808$	011	00100	000	1000
$a_4 = 9222$	010	01000	000	0110
$a_5 = 6157$	001	10000	000	1101

Хотя $r_1 = 3$, но в нашем примере начальный случайный отрезок взят длиной 4 из-за возможных переносов. Это означает, что двоичные представления имеют 16 разрядов. Максимальная длина начального сегмента в нашем примере равна 5. Но легальный пользователь сюда даже не смотрит, так что для него нет нужды знать r_1.

□

3.5. Плотные рюкзаки

Задача о рюкзаке, лежащая в основе базисного варианта криптосистемы с открытым ключом, имеет *низкую плотность*, в том смысле что компоненты рюкзачного вектора располагаются в отрезке от 1 до n очень редко. Это не так для криптосистемы, обсуждаемой в этом параграфе: лежащий в ее основе рюкзачный вектор будет *плотным* или высокой плотности. Формальные определения этих понятий будут даны ниже.

Ранее в этой главе использовалась обычная или модульная арифметика, где все числа сводились по некоторому модулю. В этом параграфе используемая арифметика будет арифметикой в конечных полях или полях Галуа. Элементарные сведения о конечных полях приведены в приложении В. Здесь мы более подробно рассмотрим некоторые понятия и докажем лемму, которая нам понадобится в этом параграфе.

Конечное поле имеет p^h элементов, где p — простое число и $h \geq 1$. Такое конечное поле часто обозначается $F(p^h)$. Опишем удобный способ представления элементов поля $F(p^h)$.

Будем рассматривать *основное поле* $F(p)$, т. е. подполе поля $F(p^h)$, состоящее из элементов $0, 1, \ldots, p-1$. В основном поле имеем обычную арифметику по модулю p. Каждый ненулевой элемент обладает обратным. Элемент α называем *алгебраическим степени* h над полем $F(p)$, если и только если α удовлетворяет в $F(p)$ уравнению $P(x) = 0$, где $P(x)$ — многочлен степени h, но не удовлетворяет никакому уравнению с многочленом меньшей степени. (Это влечет неприводимость многочлена $P(x)$). Все p^h элементов поля $F(p^h)$ могут быть представлены в виде

$$\sum c_j a^j, \quad 0 \leq c_j \leq p-1, \quad 0 \leq i \leq h-1 \ .$$

В вычислениях "коэффициенты" c_j берутся по модулю p, а степень α^i, $i \geq h$, может быть заменена на меньшую с использованием уравнения $P(\alpha) = 0$.

Например, пусть $p = 3$ и α удовлетворяет уравнению $x^2 - x - 1 = 0$. Элементы результирующего поля $F(9) = F(3^2)$ можно выразить как

$$0, 1, 2, \alpha, \alpha+1, \alpha+2, 2\alpha, 2\alpha+1, 2\alpha+2 \ .$$

В вычислениях высокие степени α заменяются на меньшие при помощи уравнения $\alpha^2 = \alpha + 1$. Так, например,

$$(\alpha+2)(2\alpha+1) = 2\alpha^2 + 5\alpha + 2 = 2\alpha + 2 + 5\alpha + 2 = \alpha + 1 \ .$$

Для данного элемента $\beta \neq 0$ из $F(p^4)$ можно рассматривать степени β^i. Ясно, что никогда не будет $\beta^i = 0$. Однако может случиться так,

что когда i пробегает значения $i = 1, 2, \ldots, p^h - 1$, то β^i пробегает все ненулевые элементы из $F(p^h)$. В таком случае будем называть β *образующей* $F^*(p^h)$ — множества (мультипликативной группы) ненулевых элементов из $F(p^h)$. Образующая может рассматриваться как основание логарифма. Вычисление логарифма элемента y из $F(p^h)$ означает вычисление такого числа a, что $\beta^a = y$. Логарифмы такого вида часто называются *дискретными логарифмами*. Считается, что их вычисление есть трудновычислимая задача. Известно, что эта задача трудна, как и задача факторизации (см. приложение Б).

Возвращаясь к примеру, запишем вначале все степени α:

i	1	2	3	4	5	6	7	8
α^i	α	$\alpha+1$	$2\alpha+1$	2	2α	$2\alpha+2$	$\alpha+2$	1

.

Из этой таблицы, кстати, видно, что α образующая. Эта таблица может быть также организована как таблица логарифмов, в которой элементы y из $F(p^h)$ перечисляются в каком-нибудь естественном (типа алфавитного) порядке:

y	1	2	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$
$\log_\alpha y$	8	4	1	2	7	5	3	6

.

Эта таблица логарифмов может применяться для умножения и деления обычным образом. Логарифмы берутся по модулю $p^h - 1$. Например,

$$\log(\alpha+2)(2\alpha+1) = \log(\alpha+2) + \log(2\alpha+1) = 10 = 2 \ ,$$

что дает $(\alpha+2)(2\alpha+1) = \alpha+1$. Аналогично,

$$\log((\alpha+1)/(2\alpha+1)) = 2 - 3 = 7 \ ,$$

что дает $(\alpha+1)/(2\alpha+1) = \alpha+2$.

Элемент $2\alpha+1$ также является образующей $F^*(9)$ с таблицей логарифмов

y	1	2	α	$\alpha+1$	$\alpha+2$	2α	$2\alpha+1$	$2\alpha+2$
$\log_{2\alpha+1} y$	8	4	3	6	5	7	1	2

.

Легко проверить, что также $\alpha+2$ и 2α являются образующими и других образующих здесь уже нет. Ясно, что β будет образующей тогда и только тогда, когда $i = p^h - 1$ есть наименьшая положительная степень, удовлетворяющая $\beta^i = 1$. Поэтому число образующих равно $\varphi(p^h - 1)$, где функция Эйлера $\varphi(x)$ дает число натуральных $i \le x$, таких, что $(i, x) = 1$. В нашем примере $\varphi(8) = 4$.

Важно отметить, что введенная выше арифметика отличается от модульной. Они совпадают только при $h = 1$.

В криптосистемах со сверхрастущим рюкзачным вектором расшифрование всегда единственно. Это следует из того, что сверхрастущие векторы являются инъективными.

Следующий вопрос относительно существования наборов с попарно различными суммами из h слагаемых рассматривался еще в 1936 г. Для заданных натуральных n и h, существует ли вектор $A = (a_1, \ldots, a_n)$ с различными нетрицательными a_i, такой, что все суммы, состоящие ровно из h компонент, не обязательно разных, попарно различны. Такие векторы A легко построить, в них компоненты a_i растут экспоненциально, например $a_i = h^{i-1}$, $1 \leq i \leq n$. Это соответствует рюкзакам низкой плотности, таким, как со сверхрастущими векторами. Но как быть в случае рюкзаков высокой плотности: можно ли построить такой вектор с a_i-ми, имеющими только полиномиальный рост относительно n. Решение, данное в [BC], представлено в следующей лемме в форме более удобной для использования в криптосистемах. Необходимо подчеркнуть, что получаемые векторы не обязательно будут инъективными, поскольку рассматриваются только суммы h компонент. На самом деле, число компонент в суммах будет $\leq h$, поскольку разрешены повторения компонент. Обозначим через p (в противоречии с нашими обычными обозначениями) число компонент в векторе A, чтобы подчеркнуть простоту числа p.

Лемма 3.8 *Пусть p — простое и $h \geq 2$ — целое число. Тогда найдется рюкзачный вектор $A = (a_1, \ldots, a_p)$, удовлетворяющий следующим условиям*

i) $1 \leq a_i \leq p^h - 1$, *для* $1 \leq i \leq p$.

ii) Пусть x_i и y_i неотрицательные целые числа и

$$(*) \qquad (x_1, \ldots, x_p) \neq (y_1, \ldots, y_p), \text{ где } \sum_{i=1}^{p} x_i = h \text{ и } \sum_{i=1}^{p} y_i = h \ .$$

Тогда

$$(**) \qquad \sum_{i=1}^{p} x_i a_i \neq \sum_{i=1}^{p} y_i a_i \ .$$

Доказательство. Рассмотрим конечное поле $F(p^h)$. Пусть α алгебраический элемент степени h над $F(p)$ и g образующая группы $F^*(p^h)$. Определим

$$a_i = \log_g(\alpha + i - 1), 1 \leq i \leq p \ .$$

Очевидно, что условие (i) выполняется, так как оно выражает просто область изменения дискретного логарифма. Для проверки условия (ii) предположим противное: найдутся x_i и y_i, удовлетворяющие (*), а вместо (**) имеем

$$(**)' \qquad \sum_{i=1}^{p} x_i a_i - \sum_{i=1}^{p} y_i a_i \ .$$

Равенство останется верным, если g возвести в степень, равную каждой части равенства $(**)'$. Принимая во внимание определение a_i, полученное из $(**)'$, равенство может быть переписано как

$$(**)'' \qquad (\alpha+0)^{x_1} \ldots (\alpha+p-1)^{x_p} = (\alpha+0)^{y_1} \ldots (\alpha+p-1)^{y_p} \ .$$

Записывая обе части $(**)''$ как многочлены относительно α, получим, что старшие степени α в обеих частях должны совпадать по условию (*). Более того, из этого же условия следует, что эта старшая степень будет равна h. Вычитая правую часть $(**)''$ из левой, получим ненулевой (из-за условия (*)) многочлен от α со старшей степенью $\leq h-1$. Таким образом, α удовлетворяет полиномиальному уравнению степени $\leq h-1$ и соответственно не может быть алгебраическим элементом степени h. Это противоречие показывает, что условие $(**)'$ не может выполняться.

□

Доказательство будет таким же и для случая, когда p есть степень простого числа. Из доказательства следует также, что условие (**) может быть заменено на более сильное

$$\sum_{i=1}^{p} x_i a_i \not\equiv \sum_{i=1}^{p} y_i a_i \pmod{p^h - 1} \ .$$

Перед тем как перейти к описанию криптосистемы, основанной на плотном рюкзаке, нам понадобится еще один вспомогательный факт. В соответствии с условиями криптосистемы исходный текст должен состоять из p-разрядных блоков, в каждом из которых имеется ровно h единиц. Произвольный двоичный набор, вообще говоря, нельзя разделить на такие блоки. Однако интуитивно ясно, что блоки можно вначале закодировать более длинными двоичными блоками, уже удовлетворяющими требуемому условию. Это будет показано в следующей лемме. Из многих возможных конструкций в доказательстве мы выбрали самую простую.

Лемма 3.9 *Рассмотрим натуральное число* $h \geq 3$ *и* $h < n$. *Тогда существует вложение множества всех двоичных наборов длины* $[\log_2 \binom{n}{h}]$ *во множество всех таких двоичных наборов длины* n, *у которых число единиц равно* h.

Доказательство. Будем рассматривать двоичные наборы длины $[\log_2 \binom{n}{h}]$ как двоичные представления чисел a. Все двоичные наборы длины n, содержащие по h единиц каждый, запишем в алфавитном порядке, при котором 0 предшествует 1. Так, первым набором согласно этому упорядочению будет набор, в котором все единицы стоят в конце, а в последнем наборе — в начале. Отобразим двоичный набор, представляющий число a в $(a+1)$-й набор в построенном упорядоченном списке. Ясно, что это отображение будет вложением. Более того, мы не выйдем за пределы списка, поскольку в нем $\binom{n}{h}$ элементов, а двоичных наборов длины x, где $x = [\log_2 \binom{n}{h}]$, всего 2^x, что не превосходит $\binom{n}{h}$.

□

В качестве примера рассмотрим $n = 5,\ h = 2$. Тогда

$$[\log_2 \binom{n}{h}] = 3$$

и, следовательно, мы можем кодировать двоичные наборы длины 3. Кодирование, используемое в вышеприведенном доказательстве, показано ниже. В первой колонке перечисляются наборы длины 3, а во второй — соответствующие наборы длины 5 с ровно двумя единицами.

0 0 0	0 0 0 1 1
0 0 1	0 0 1 0 1
0 1 0	0 0 1 1 0
0 1 1	0 1 0 0 1
1 0 0	0 1 0 1 0
1 0 1	0 1 1 0 0
1 1 0	1 0 0 0 1
1 1 1	1 0 0 1 0

Отметим, что наборы 10100 и 11000 при этом не используются. Выберем теперь $n = 7,\ h = 2$. Используя приведенную технику, мы могли бы закодировать только все 16 двоичных наборов длины 4. Однако двоичных наборов длины 7 с двумя единицами будет 21. Ниже приводится кодирование 21 буквы английского алфавита.

A	0 0 0 0 0 1 1	G	0 0 1 0 0 0 1
B	0 0 0 0 1 0 1	H	0 0 1 0 0 1 0
C	0 0 0 0 1 1 0	I	0 0 1 0 1 0 0
D	0 0 0 1 0 0 1	K	0 0 1 1 0 0 0
E	0 0 0 1 0 1 0	L	0 1 0 0 0 0 1
F	0 0 0 1 1 0 0	M	0 1 0 0 0 1 0

N	0 1 0 0 1 0 0	T	1 0 0 0 1 0 0
O	0 1 0 1 0 0 0	U	1 0 0 1 0 0 0
P	0 1 1 0 0 0 0	V	1 0 1 0 0 0 0
R	1 0 0 0 0 0 1	W	1 1 0 0 0 0 0
S	1 0 0 0 0 1 0		

Теперь все готово для описания криптосистемы. Мы сделаем это с точки зрения создателя системы, который будет также и легальным получателем сообщений.

Сделаем вначале осмысленный выбор степени простого числа, т.е. p и $h \leq p$ так, чтобы можно было бы эффективно вычислять дискретный логарифм в конечном поле $F(p^h)$. (Некоторые алгоритмы действительно хорошо работают, например, в случае когда $p^h - 1$ имеет только небольшие простые делителя.)

Далее, выбираем алгебраический элемент α степени h над $F(p)$, а также образующую g группы $F^*(p^h)$. Для выбора пары (α, g) имеется много возможностей. Конечно, элемент α не обязательно должен быть тем же, что и при задании элементов поля $F(p^h)$. Далее, для простоты мы будем предполагать, что это, однако, так.

Вычисляем

$$(*) \qquad a_i = \log_g(\alpha + i - 1), \quad 1 \leq i \leq p \ .$$

Это основной шаг в построении системы.

Перемешиваем числа a_i посредством случайно выбранной перестановки π чисел $1, \ldots, p$ и добавим по модулю $p^h - 1$ к результату произвольно выбранное $d, 0 \leq d \leq p^h - 2$. Пусть $B = (b_1, \ldots, b_p)$ обозначает полученный вектор. (Перестановка π и сдвиг d, вообще говоря, не являются существенными. Мы включили их сюда для того, чтобы криптосистема совпадала с описанной в [Cho].)

Открытый ключ зашифрования составляют B, p и h. *Секретную лазейку* образуют α, g, π и d.

Пусть теперь C есть двоичный набор длины p, в котором ровно h единиц. Вектор C, рассматриваемый как вектор-столбец, *зашифровывается* как наименьший положительный остаток от $BC \pmod{p^h - 1}$.

Если h близко к p, то можно зашифровать аналогичным способом векторы C размерности p, у которых сумма компонент равна h. Однозначность расшифрования, вытекающая из леммы 3.8, остается в силе.

Расшифрование для *легального получателя*, знающего лазейку, выполняется следующим образом. Вычитая по модулю $p^h - 1$ число hd из криптотекста, получаем число y.

Вычисляем в поле $F(p^h)$ степень g^y. Она будет представлять собой многочлен относительно α степени не выше $h-1$. С другой стороны, α удовлетворяет уравнению $\alpha^h = r(\alpha)$, где $r(\alpha)$ — многочлен степени не выше $h-1$. Многочлен

$$s(\alpha) = \alpha^h + g^y - r(\alpha)$$

раскладывается над $F(p)$ на линейные множители, поскольку $s(\alpha)$ есть произведение степеней g, где каждая степень имеет вид (*). Вычитание hd из криптотекста нейтрализует добавление случайного шума d. Линейные множители находятся подстановкой чисел $0, 1, \ldots, p-1$. Таким образом, получим

$$s(\alpha) = (\alpha + i_1 - 1) \ldots (\alpha + i_h - 1) \ .$$

Места для единиц в исходном тексте определяются после применения обратной перестановки π^{-1} к числам $i_1, \ldots, i_h$. Лемма 3.8 гарантирует, что результат процедуры расшифрования всегда будет однозначным.

Криптоаналитик сталкивается, по существу, с общей (модульной) задачей о рюкзаке. Алгоритмы же типа рассмотренного в примере 3.2 не применимы, поскольку в них предполагаются рюкзаки низкой плотности. Возможно, в этом случае можно развить аналог теории достижимости, не зависящий от плотности. Криптоанализ, когда что-либо известно, обсуждается в задаче 44. В общем случае *плотность* рюкзачного вектора $A = (a_1, \ldots, a_n)$ определяется как

$$d(A) = \frac{n}{\log_2 \max A} \ .$$

Для сверхрастущего вектора A имеем $a_n \geq 2^{n-1}$ и соответственно $d(A) \leq n/(n-1)$. Обычно же плотность гораздо ниже, чем $n/(n-1)$ в сверхрастущем случае. Поскольку $\max A \geq n$, для любого рюкзачного вектора всегда $d(A) \leq n/\log_2 n$. Например, для $A = (1, 2, 3, \ldots, 128)$ имеем $d(A) = 128/7 = 18.2857$. Хотя мы имеем верхнюю оценку для $d(a)$, нижней положительной оценки в терминах n нет.

Пример 3.7. В следующих двух примерах никакого тасования с помощью π и d не выполняется. Эти примеры так малы, что легко осуществляется непосредственный криптоанализ (т.е. решение задачи о

рюкзаке). Без перемешивания основная идея криптосистем, связанных с плотными рюкзаками, становится более наглядной.

Рассмотрим вначале пример поля $F(9)$, уже представленного выше, где α удовлетворяет уравнению $x^2 = x + 1$, неприводимому над $F(3)$. Выберем образующую $2\alpha + 1$, рассмотренную в начале параграфа. Так как логарифмы элементов $\alpha, \alpha + 1$ и $\alpha + 2$ есть числа 3, 6 и 5, то мы получим открытый ключ зашифрования $A = (3, 6, 5)$.

Исходные тексты есть векторы размерности 3 с суммой компонент равной 2. Рассмотрим исходные тексты $(2, 0, 0)$ и $(0, 1, 1)$. Они шифруются как числа 6 и 3. (Напомним, что мы работаем с модулем $p^h - 1 = 8$.) Для расшифрования легальный получатель вычисляет степени

$$(2\alpha + 1)^6 = \alpha + 1 \text{ и } (2\alpha + 1)^3 = \alpha \ .$$

(Здесь удобно воспользоваться таблицей логарифмов.) Когда $\alpha^2 - \alpha - 1$ прибавляется к обеим степеням, то возникают многочлены

$$\alpha^2 \text{ и } \alpha^2 - 1 = (\alpha + 1)(\alpha + 2) \ ,$$

немедленно дающие первоначальные исходные тексты $(2, 0, 0)$ и $(0, 1, 1)$.

Мимоходом отметим, что в лемме 3.8 нельзя заменить в (*) два знака равенства на знаки неравенства $\leq$. В противном случае, вектор $A = (3, 6, 5)$, построенный по лемме, и два вектора $(2, 0, 0)$ и $(0, 1, 0)$ образовывали бы контрпример. (Лемма 3.8 приведена в неверной форме, см. например, [Cho].)

Рассмотрим все тот же пример, но перемешаем теперь A циклической перестановкой $\pi : 1 \to 2, 2 \to 3, 3 \to 1$ с добавлением шума $d = 7$. В результате получим вектор $B = (5, 4, 2)$. Вектор B, $p = 3$ и $h = 2$ составляют открытый ключ зашифрования, в то время как π, d, многочлен $x^2 - x - 1$, определяющий α, и образующая $2\alpha + 1$ формируют секретную лазейку. Исходный текст $(0, 1, 1)$ шифруется как 6. Легальный получатель позаботится вначале об устранении шума, вычислив наименьший положительный остаток 8 от $6 - 2 \cdot 7 \pmod 8$. Далее вычисляется

$$\alpha^2 + (2\alpha + 1)^8 - \alpha - 1 = \alpha^2 - \alpha = \alpha(\alpha + 2) \ .$$

Это дает вектор $(1, 0, 1)$, из которого первоначальный исходный текст $(0, 1, 1)$ получается обратной перестановкой π^{-1}: $1 \to 3$, $2 \to 1$, $3 \to 2$.

В нашем последнем примере используется конечное поле $F(64) = F(2^6)$. Чтобы сделать пример читаемым, мы не будем рассматривать его в самом общем виде. Итак, $p = 2$, $h = 6$. Это противоречит сделанному ранее соглашению $h < p$, однако оставляет в силе все используемые аргументы. Мы могли бы также взять, например, $p = 8$, $h = 2$.

Многочлен $x^6 - x - 1$ неприводим над $F(2)$. (Это следует из того, что ни 0, ни 1 не обращают его в 0.) Следовательно, $F(2^6)$ может быть представлено через корень α этого многочлена. Более конкретно, 64 элемента поля $F(2^6)$ могут быть выражены в виде

$$\sum_{i=1}^{6} x_i \alpha^{6-i} ,$$

где $x_i = 0, 1$. Будем представлять элементы просто как двоичные наборы длины 6. Так, $\alpha^5+\alpha^4+\alpha^3+\alpha^2+\alpha+1, \alpha^4+\alpha^2+\alpha$ и 1 представляются наборами 111111, 010110 и 000001 соответственно. Выбираем α также и как образующую $F(2^6)$. Тогда таблица логарифмов выглядит следующим образом. Каждый элемент поля $F(2^6)$ приводится, так же и как двоичный набор.

Элемент	Логарифм	Элемент	Логарифм
1=000001	63	33=100001	62
2=000010	1	34=100010	25
3=000011	6	35=100011	11
4=000100	2	36=100100	34
5=000101	12	37=100101	31
6=000110	7	38=100110	17
7=000111	26	39=100111	47
8=001000	3	40=101000	15
9=001001	32	41=101001	23
10=001010	13	42=101010	53
11=001011	35	43=101011	51
12=001100	8	44=101100	37
13=001101	48	45=101101	44
14=001110	27	46=101110	55
15=001111	18	47=101111	40
16=010000	4	48=110000	10
17=010001	24	49=110001	61
18=010010	33	50=110010	46
19=010011	16	51=110011	30
20=010100	14	52=110100	50
21=010101	52	53=110101	22
22=010110	36	54=110110	39
23=010111	54	55=110111	43
24=011000	9	56=111000	29
25=011001	45	57=111001	60
26=011010	49	58=111010	42
27=011011	38	59=111011	21
28=011100	28	60=111100	20
29=011101	41	61=111101	59
30=011110	19	62=111110	57
31=011111	56	63=111111	58
32=100000	5		

Для шума $d = 60$ получается открытый вектор $B = (61, 3)$. (На самом деле это не есть рюкзачный вектор, поскольку $p = 2$.) Исходными текстами являются векторы (x, y), где $x + y = 6$. Используя открытый ключ зашифрования B, $p = 2$, $h = 6$ исходный текст $(1, 5)$ шифруется как 13. Легальный получатель вычисляет число

$$(13 - 6 \cdot 60, \mathrm{mod} 63) = 31 \ .$$

Следующие равенства немедленно получаются из таблицы логарифмов

$$\alpha^3 1 = \alpha^5 + \alpha^2 + 1,\ \alpha^6 + \alpha^3 1 - \alpha - 1 == \alpha^6 + \alpha^5 + \alpha^2 + \alpha = \alpha(\alpha+1)^5\ ,$$

из которых извлекается исходный текст $(1,5)$. Аналогично, исходный текст $(6,0)$ шифруется как 51. Удаляя шум, законный получатель получает число 6. Это дает многочлен α^6, соответствующий исходному тексту $(6,0)$. Читатель может взять и другие примеры, а также рассмотреть модификации с другими значениями p (например, $p = 8$, $h = 2$) и применить некоторую перестановку π.

□

Глава 4

Криптосистема RSA

4.1. Легальный мир

Наиболее широко используемой и проверенной криптосистемой с открытым ключом, которая была придумана Ривестом, Шамиром и Эдлеманом (Rivest, Shamir, Adleman), является система RSA. Она основана на удивительно простой теоретико-числовой (можно сказать, даже арифметической) идее и еще в состоянии сопротивляться всем криптоаналитическим атакам. Идея состоит в искусном использовании того факта, что легко перемножить два больших простых числа, однако крайне трудно разложить на множители их произведение. Таким образом, произведение может быть открыто и использовано в качестве ключа зашифрования. Исходные простые числа не могут быть восстановлены из их произведения. С другой стороны, эти простые числа необходимы при расшифровании. Следовательно, мы имеем прекрасный каркас для криптосистемы с открытым ключом. Более того, детали этой системы могут быть объяснены очень быстро, вот почему мы называем систему “изумительно простой”.

Этот параграф связан с RSA с позиции легальных пользователей. Мы обсудим разработку системы, зашифрование и расшифрование. Также будут представлены оба аспекта употребления — секретность и идентификация. В параграфе 4.2 говорится о некоторых простых фактах и мерах предосторожности при разработке системы. В следующих двух параграфах показана взаимосвязь RSA с задачей факторизации. К примеру, в параграфе 4.3 представлены тесты проверки чисел на простоту. В параграфе 4.4 обсуждается криптоанализ без факторизации, т.е. вскрытие RSA с помощью некоторой информации, отличной

от простых чисел p и q. В параграфе 4.5 приводится роль частичной информации: если мы в состоянии найти определенные факты об исходном тексте, мы в состоянии вскрыть всю систему. Это означает, что RSA надежна так же, как и отдельные ее части. В этом смысле параграфы 4.2 – 4.5 фокусируют внимание на взаимодействии легального и нелегального миров, т.е. между разработчиком системы и криптоаналитиком. Последний параграф 4.6 знакомит с некоторыми системами, связанными с RSA; это представление будет продолжено далее в гл. 5.

Отметим, что не существует формального доказательства любого из фактов: (1) факторизация трудновычислима или трудновычислима в специальном смысле, используемом в RSA, и (2) факторизация необходима для криптоанализа RSA, т.е. не существует криптоаналитических методов, не использующих факторизацию. Имеется множество эмпирических подтверждений для обоих пунктов (1) и (2).

Теперь рассмотрим алгоритм RSA более подробно. Пусть p и q — два различных больших случайно выбранных простых числа (имеющих обычно 100 разрядов в их десятичном представлении). Обозначим

$$n = pq \quad \text{и} \quad \varphi(n) = (p-1)(q-1) \ .$$

(Здесь φ — функция Эйлера, которая встречалась ранее в параграфе 3.5.) Случайно выберем большое число $d > 1$, такое, что $(d, \varphi(n)) = 1$, и вычислим e, $1 < e < \varphi(n)$, удовлетворяющее сравнению

$$ed \equiv 1 \pmod{\varphi(n)} \ .$$

Числа n, e и d называются *модулем*, *экспонентой зашифрования* и *расшифрования* соответственно. Числа n и e образуют *открытый ключ зашифрования*, тогда как оставшиеся числа p, q, $\varphi(n)$ и d формируют *секретную лазейку*. Очевидно, что секретная лазейка включает в себя взаимозависимые величины. К примеру, зная p, нетрудно вычислить оставшиеся три величины.

При *зашифровании* исходный текст возводится в степень e по модулю n. При *расшифровании* криптотекст возводится в степень d по модулю n.

Более детально: потребуем, чтобы исходный текст кодировался десятичным числом (аналогично можно использовать и двоичное представление). Данное число затем делится на блоки подходящего размера. Блоки шифруются отдельно. Их подходящий размер определяется как единственное целое число i, удовлетворяющее неравенствам $10^{i-1} < n < 10^i$. В некоторых случаях в качестве размера блоков можно выбрать число $i-1$, однако, если важна однозначность расшифрования, нужно быть уверенным в том, что каждому блоку соответствует число,

меньшее n. Например, в приводимой ниже задаче 4.1 $n = 2773$, откуда следует, что размер блока равен 4. Числа $1000 + 2773j$ шифруются одинаково для $j = 0, 1, 2, 3$. Однако в исходном тексте из примера 4.1 для j возможно только нулевое значение.

Если w является блоком исходного текста, а c — соответствующим блоком криптотекста, то зашифрование может быть описано в терминах следующего сравнения:

$$c = (w^e, \mod n) .$$

Теперь покажем корректность расшифрования.

Лемма 4.1 *Для w и c, определенных выше,*

$$(*) \qquad w \equiv c^d \pmod{n} .$$

Следовательно, если расшифрование однозначно, то $w = (c^d, \mod n)$.

Доказательство. В силу выбора d существует положительное целое число j, такое, что $ed = j\varphi(n) + 1$. Потребуем сначала, чтобы ни p, ни q не делили w. По теореме Эйлера (см. приложение Б) $w^{\varphi(n)} \equiv 1 \pmod{n}$, откуда $w^{ed-1} \equiv 1 \pmod{n}$. Следовательно,

$$c^d \equiv (w^e)^d \equiv w \pmod{n} .$$

Если в точности одно из p и q, скажем p, делит w, то $w^{q-1} \equiv 1 \pmod{q}$. Поэтому

$$w^{\varphi(n)} \equiv 1 \pmod{q}, \quad w^{j\varphi(n)} \equiv 1 \pmod{q}, \quad w^{ed} \equiv w \pmod{q} .$$

Так как последнее сравнение верно и по модулю p, получаем (*). Если и p, и q делят w, имеем $w^{ed} \equiv w \pmod{n}$, откуда, как и ранее, следует (*).

□

Рассмотрим вновь $n = 2773$. Если мы выбираем размер блока 4, может случиться так, что расшифрование приведет обратно не к оригинальному исходному тексту w, к примеру, когда $w = 3773$.

Теперь обсудим разработку криптосистемы, а именно как генерируются различные ее части. В целом, когда мы говорим, что взято случайное число или же мы выбираем что-нибудь случайно, то мы используем генератор случайных чисел, например компьютерную программу, генерирующую такую последовательность разрядов, чтобы у

нее было как можно больше статистических свойств случайной последовательности. Мы не обсуждаем здесь детали генераторов случайных чисел. Для определения двух огромных случайных простых чисел p и q произвольно выбирается нечетное целое число r подходящего размера (скажем, 100 разрядов) и проверяется на простоту. Тесты для проверки чисел на простоту описываются в параграфе 4.3. В случае отрицательного ответа проверяется $r+2$ и т. д. По теореме о простых числах существует примерно

$$\frac{10^{100}}{\ln 10^{100}} - \frac{10^{99}}{\ln 10^{99}}$$

100-разрядных простых чисел. (Здесь ln означает натуральный логарифм.) Если это число сравнить с числом $(10^{100} - 10^{99})/2$ всех 100-разрядных нечетных чисел, видно, что вероятность успеха для конкретного теста приблизительно равна 0.00868.

После того как p и q выбраны, кандидаты для d проверяются с помощью алгоритма Евклида. Когда d удовлетворяет условию $(d, \varphi(n)) = 1$, цепочка равенств, получаемых из алгоритма Евклида, дает тут же и e.

Операцией, необходимой при зашифровании и расшифровании, является *модульное возведение в степень*, т. е. вычисление $(a^r, \mod n)$. Это можно сделать намного быстрее, чем при помощи повторяющегося умножения a на себя. Представляемый нами метод называется *методом последовательного возведения в квадрат*. После каждого возведения в квадрат результат сводится по модулю n. При этом никогда не возникают числа большие n^2.

Поясним это более подробно. Рассмотрим двоичное представление r

$$r = \sum_{j=0}^{k} x_j 2^j, \quad x_j = 0, 1; \quad k = [\log_2 r] + 1 \ .$$

Предположим, что мы знаем все числа

$$(*) \qquad (a^{2^j}, \ \mathrm{mod}\, n), \quad 0 \le j \le k \ ,$$

тогда $(a^r, \mod n)$ может быть вычислено с помощью не более $k-1$ произведений и сведением каждого произведения по модулю n. Таким образом, достаточно вычислить числа (*), которые требуют k модульных возведения в квадрат и дополнительно не более $k-1$ модульных произведений. Это означает, что вычисляется не более $2k-1$ произведений с обоими множителями, меньшими чем n, и сведением произведений по модулю n. Если r является большим и известно $\varphi(n)$, то r может быть вначале взято по модулю $\varphi(n)$.

К примеру, вычисляя $(7^{83},\ \mathrm{mod}\ 61)$, замечаем, что $7^{60} \equiv 1\ (\mathrm{mod}\ 61)$. Следовательно, мы можем вычислить $(7^{23},\ \mathrm{mod}\ 61)$.

С помощью возведения в квадрат мы получаем степени семерки, где показатель в свою очередь является степенью двойки:

j	0	1	2	3	4
7^{2^j}	7	49	22	54	16

Так как $23 = 10111$, мы получаем желаемый результат следующим образом:

$$(7^{23},\ \mathrm{mod}\ 61) = (16\,(22\,(49 \cdot 7)),\ \mathrm{mod}\ 61) = 17\ .$$

Иногда можно найти результат намного быстрее. Это имеет существенное значение при малой мощности вычислительных средств. К примеру, при вычислении $(191^{239},\ \mathrm{mod}\ 323)$ вначале отмечаем, что

$$191^4 \equiv 1\ (\mathrm{mod}\ 323),\ \text{поэтому}\ 191^{236} \equiv 1\ (\mathrm{mod}\ 323)\ .$$

Так как $(191^3,\ \mathrm{mod}\ 323) = 115$, ответ на первоначальный вопрос также будет 115.

Мы уже рассмотрели модуль $n = 2773$ и еще вернемся к нему в примере 4.1. Вычисляя $(1920^{17},\ \mathrm{mod}\,2773)$, рассмотрим сначала степени двойки как показатели степени:

j	0	1	2	3	4
1920^{2^j}	1920	1083	2683	2554	820

Мы заключаем, что

$$(1920^{17},\ \mathrm{mod}\,2773) = (1920 \cdot 820,\ \mathrm{mod}\,2773) = 2109\ .$$

Так как $17^{-1} = 157\ (\mathrm{mod}\ 2668)$, мы можем проверить результат, аналогично вычисляя

$$(2109^{157},\ \mathrm{mod}\,2773) = 1920\ .$$

Заметим, что $2773 = 47 \cdot 59$ и $\varphi(2773) = 2668$.

После того как мы представим в параграфе 4.3 быстрые стохастические алгоритмы для проверки чисел на простоту, мы сможем сделать вывод, что все вычисления, которые необходимы при создании криптосистемы, как при зашифровании и легальном расшифровании, могут быть получены за низкополиномиальное время. Дополнительно отметим, что легальные операции в DES приблизительно в 1000 раз быстрее, чем в RSA.

Пример 4.1. Рассмотрим три иллюстрации с растущими модулями. Все они будут читаемыми, хотя даже наибольший из них нереалистичен с точки зрения практической безопасности.

Сначала возьмем $p = 5$, $q = 11$, $n = 55$, $\varphi(n) = 40$, $e = 7$, $d = 23$. Пусть теперь исходные сообщения являются целыми числами отрезка $[1, 54]$. Более того, мы хотим исключить числа, для которых наибольший общий делитель с 55 превышает 1, т.е. числа, делящиеся на 5 или 11. В общем случае, если $(w, n) > 1$ для некоторого исходного текста w, то можно разложить n на множители путем вычисления наибольшего общего делителя n и зашифрованной версии w. Конечно, в этом примере мы можем факторизовать n любым образом. В целом вероятность того, что исходный текст имеет общий нетривиальный множитель с n, меньше чем $1/p + 1/q$. Поэтому при больших p и q эта вероятность ничтожна.

В примере легко даже вручную записать полную таблицу зашифрования.

Исходный текст	Криптотекст	Исходный текст	Криптотекст
1	1	28	52
2	18	29	39
3	42	31	26
4	49	32	43
6	41	34	34
7	28	36	31
8	2	37	38
9	4	38	47
12	23	39	19
13	7	41	46
14	9	42	48
16	36	43	32
17	8	46	51
18	17	47	53
19	24	48	27
21	21	49	14
23	12	51	6
24	29	52	13
26	16	53	37
27	3	54	54

Эта таблица может быть преобразована к полной таблице расшифрования.

Криптотекст	Исходный текст	Криптотекст	Исходный текст
1	1	28	7
2	8	29	24
3	27	31	36
4	9	32	43
6	51	34	34
7	13	36	16
8	17	37	53
9	14	38	37
12	23	39	29
13	52	41	6
14	49	42	3
16	26	43	32
17	18	46	41
18	2	47	38
19	39	48	42
21	21	49	4
23	12	51	46
24	19	52	28
26	31	53	47
27	48	54	54

Следующий важный факт наглядно продемонстрирован на этом примере. Криптография с открытым ключом никогда не работает с маленькими пространствами исходных сообщений. Криптоаналитик может построить уже на предварительной стадии полную таблицу зашифрования простым зашифрованием всевозможных исходных сообщений и переупорядочить результирующие криптотексты в подходящем алфавитном порядке.

В качестве второй иллюстрации рассмотрим $p = 47$, $q = 59$, $n = 2773$, $\varphi(n) = 2668$, $e = 17$, $d = 157$. Теперь исходный текст, закодированный как последовательность десятичных разрядов, делится на блоки из четырех разрядов. Как мы говорили выше, это может привести к неоднозначности в процессе расшифрования. Однако неопределенности не будет, если оригинальный исходный текст записан с использованием 26 букв английского алфавита, в этом случае наибольшее четырехразрядное число равно 2626.

Теперь мы, записывая исходное сообщение для повышения безопасности на финском языке, зашифруем текст SAUNOIN TAAS (I took a sauna bath again). Числовые коды пар с пробелом, закодированным 00, дают следующую таблицу:

Блок исходного текста	SA	UN	OI	N-	TA	AS
Код	1901	2114	1509	1400	2001	0119

Модульное возведение в степень, необходимое для зашифрования, осуществляется путем последовательного модульного возведения в квадрат, как видно из следующей таблицы:

Исходный текст w	1901	2114	1509	1400	2001	0119
w^2	582	1693	448	2262	2562	296
w^4	418	1740	1048	459	153	1653
w^8	25	2257	196	2706	1225	1004
w^{16}	625	48	2367	1716	432	1417
Криптотекст w^{17}	1281	1644	179	982	2029	2243

Результат может быть проверен возведением криптотекста c в степень 157. К примеру, если $c = 1644$, мы получаем

$$c^2 = 1834, \quad c^4 = 2680, \quad c^8 = 330, \quad c^{16} = 753, \quad c^{32} = 1317,$$
$$c^{64} = 1364, \quad c^{128} = 2586, \quad c^{144} = 612, \quad c^{152} = 2304,$$
$$c^{156} = 2022, \quad c^{157} = 2114.$$

В качестве нашей последней иллюстрации рассмотрим последовательность чисел:

$$\begin{aligned} p &= 3336670033\ , \\ q &= 9876543211\ , \\ n &= 32954765761773295963\ , \\ \varphi(n) &= 32954765748560082720\ , \\ e &= 1031\ , \\ d &= 31963885304131991\ . \end{aligned}$$

Блоки исходного текста будут теперь состоять из 20 разрядов. Неопределенности не будет, если блоки исходного текста получаются из английского текста числовым кодированием, откуда следует, что все блоки начинаются с 0, 1 или 2.

Пусть шифруется следующее исходное сообщение: “Sauna stoves are either preheated or continuously heated. Preheated means that the stove is not heated during the actual bathing. A smoke sauna is a special type of a preheated sauna. There is no chimney but smoke goes out through holes in the walls and roof”.

В зашифровании на следующей странице опущены все знаки препинания и нет отличия между строчными и прописными буквами. Мы указываем деление на блоки и числовые коды в обычном смысле.

□

Теперь вернемся к общему случаю. Система RSA также может быть рассмотрена согласно общим принципам построения криптосистем с открытым ключом, представленных в параграфе 2.2. Начальная постановка здесь не так ясна, как, к примеру, для криптосистем на основе задачи об укладке рюкзака. В качестве трудной задачи P можно выбрать факторизацию n, когда известно, что n — это произведение двух простых чисел. Легкой подзадачей P_{easy} является задача P с дополнительным знанием $\varphi(n)$. Перемешанная версия P_{easy} есть просто сама P. Или в качестве трудной задачи P можно выбрать решение сравнения

$$x^e \equiv c \pmod{n} ,$$

когда известна тройка (e, c, n), образующая RSA. P_{easy} в этом случае есть задача P с дополнительным значением одной из величин $\varphi(n)$, d, p, q. Очевидны следующие два пункта (1) и (2), хотя они порождают много недоразумений.

(1) Нет противоречия между тем, что для заданного числа n можно определить, является ли оно составным или нет, и тем, что n не может быть факторизовано. Первый факт требуется при разработке RSA-систем, тогда как факторизация n вскрывает криптосистему. Действительно, существует много результатов в форме "если n — простое, то выполняется условие $C(n)$". Следовательно, если мы заметим, что $C(n)$ не выполняется, то n — составное, но это еще не значит, что мы можем факторизовать n.

(2) Для функций действительных переменных не существует разницы, с точки зрения сложности, между возведением в степень и вычислением логарифмов. В дискретном случае модульное возведение в степень является легким, в то время как нахождение логарифмов трудновычислимо. Последняя задача будет рассмотрена в параграфе 4.6.

Задачи *идентификации* и *цифровой подписи* уже обсуждались в параграфе 2.3. Подписи указывают на пользователя. Пусть e_A, d_A, n_A — экспоненты зашифрования, расшифрования и модуль, используемые A. Потребуем вначале, чтобы при передаче сообщений необходимой была только подпись, а не секретность. Тогда A посылает пару $(w, D_A(w))$, где

$$D_A(w) = (w^{d_A}, \bmod n_A) .$$

Получатель может проверить подпись, применяя опубликованную экспоненту зашифрования A e_A. Так как только A обладает d_A, никто другой не может подписать сообщение w.

Однако противник может выбрать число c, вычислить

$$E_A(c) = (c^{e_A}, \bmod n_A)$$

и с успехом утверждать, что c — подпись A в сообщении $E_A(c)$. Этот метод атаки может быть использован только для нахождения подписей непредсказуемых сообщений: только A может подписать выбранное сообщение. Такие непредсказуемые сообщения будут иметь смысл с очень невысокой вероятностью, например, когда исходные сообщения получаются числовым кодированием некоторого естественного языка. Тогда *избыточность* в сообщениях высока, и только очень маленькие порции блоков определенного размера являются числовыми кодами частей осмысленного исходного текста. В дополнение к *степени* неопределенности важен также *тип* неопределенности исходных текстов. В частности, ни инверсия осмысленного сообщения, ни произведение двух осмысленных сообщений не будет осмысленным. Иначе перехватчик, зная подпись s_i A в сообщении w_i, $i = 1, 2$, может подписать с правильной подписью A сообщение $(w_1 w_2, \mod n_A)$ и $(w_1^{-1}, \mod n_A)$, используя $(s_1 s_2, \mod n_A)$ и $(s_1^{-1}, \mod n_A)$.

Рассмотрим первую иллюстрацию примера 4.1. Подписями для сообщений 12 и 29 являются 23 и 24 соответственно. Очевидно, что

$$(12 \cdot 29, \mod 55) = 18 \quad \text{и} \quad (29^{-1}, \mod 55) = 19$$

может быть затем подписано, используя

$$(23 \cdot 24, \mod 55) = 2 \quad \text{и} \quad (24^{-1}, \mod 55) = 39 \ .$$

Для построения новых подписей требуются только n_A и подписи для w_i. Этот метод подделки приведет к произведению более чем двух множителей. Так как, очевидно, d_A всегда нечетно, мы можем подписать также $(-w_1, \mod n_A)$, используя $(-s_1, \mod n_A)$.

Потребуем теперь, чтобы при посылке сообщения требовались и подпись, и секретная передача: посылая подписанное сообщение к B, A вначале подписывает его, используя (d_A, n_A), а затем зашифровывает результат, применяя пару (e_B, n_B). B сначала расшифровывает текст, используя экспоненту расшифрования d_B, после чего оригинальное сообщение может быть получено применением открытого ключа зашифрования e_A. Присутствие d_A в сообщении гарантирует, что оно было послано от A. Как и ранее, надо быть осторожным, потому что возможна подделка подписи в непредсказуемых сообщениях.

Существует также и другая сложность, возникающая из того, что A и B используют разные модули. Пусть $n_A > n_B$. Тогда $D_A(w)$ не обязательно лежит в интервале $[1, n_B - 1]$ и сведение по модулю n_B, которое будет обязательно выполняться при легальном расшифровании, очень трудно. Существует два пути преодоления этой сложности.

(1) Все пользователи договариваются об общем *пороге* t. Каждый пользователь A выбирает два ключа RSA — один для подписи, другой для зашифрования. Части обозначаются s и e соответственно. Каждый пользователь A заботится о том, чтобы $n_A^s < t < n_A^e$. Трудности, описанные выше, не появляются, если A посылает сообщение w к B в виде

$$E_B^e(D_A^s(w)) \ .$$

(2) Можно также избежать порогового числа, если сообщения от A к B посылаются в виде $E_B(D_A(w))$ или $D_A(E_B(w))$, в зависимости от того, какое из неравенств $n_A < n_B$, $n_B < n_A$ выполняется.

4.2. Нападение и защита

В предыдущем параграфе мы уже обсудили, как противодействовать атаке с подделкой подписей. Существует много криптоаналитических атак против криптосистем RSA, но ни одна из них не кажется серьезной. В этом параграфе мы обсудим некоторые типичные атаки и обратим внимание на некоторые другие аспекты, касающиеся предотвращения очевидных атак.

Рассмотрим вначале выбор p и q. Они должны быть случайными простыми числами и не содержаться ни в одной из известных таблиц простых чисел. При факторизации можно всегда проверить всю таблицу или перебрать последовательность простых чисел специфического вида. Эти два простых числа также не должны быть близкими друг к другу. В противном случае (при $p > q$) $(p-q)/2$ мало, а $(p+q)/2$ только немного больше, чем $\sqrt{n}$. Более того

$$\frac{(p+q)^2}{4} - n = \frac{(p-q)^2}{4}$$

и, следовательно, левая сторона равенства образует полный квадрат. Факторизуя n, проверяем целые числа $x > \sqrt{n}$ до тех пор, пока не найдем такое, что $x^2 - n$ есть полный квадрат, скажем y^2. Тогда $p = x + y$ и $q = x - y$.

К примеру, для $n = 97343$ мы имеем $\sqrt{n} = 311998$. Теперь $312^2 - n = 1$, которое прямо дает $p = 313$ и $q = 311$. В целом рекомендуется, чтобы двоичное представление p и q различалось по длине на несколько битов.

При выборе p и q также должно быть рассмотрено и $\varphi(n)$. Оба числа $p-1$ и $q-1$ четны, из чего следует, что $\varphi(n)$ делится на 4. Потребуем, чтобы $(p-1, q-1)$ было большим, и поэтому наименьшее общее кратное

$p-1$ и $q-1$ мало по сравнению с $\varphi(n)$. Тогда любая инверсия e по модулю u будет работать как экспонента расшифрования. Так как в этом случае намного легче найти d простой проверкой, $p-1$ и $q-1$ не должны иметь большой общий делитель.

Вырожденным является случай, когда одно из $p-1$ или $q-1$, скажем $q-1$, делит другое. Тогда достаточно рассмотреть инверсии e по модулю $p-1$. Вновь рассмотрим пример. Пусть $n = 11041$ и $e = 4013$. Любая инверсия 4013 по модулю 180 может быть использована как экспонента расшифрования, так как наименьшее общее кратное $p-1$ и $q-1$ будет равно 180. Таким образом, мы получаем $d = 17$.

Разработчик криптосистемы также должен избегать ситуации, когда в разложении $\varphi(n)$ на множители участвуют только очень маленькие простые числа. Пусть все простые множители r $\varphi(n)$ меньше некоторого целого k. Так как $[\log_r n]$ является наибольшим показателем степени множителя r, который, вероятно, может делить $\varphi(n)$, то можно перебрать всех кандидатов v для $\varphi(n)$ и проверить криптотекст возведением в степень $(v+1)/e$, обеспечивая при этом, чтобы показатель был целым числом.

Путем преодоления обеих трудностей, касающихся $\varphi(n)$, является рассмотрение только "безопасных" простых чисел. По определению, простое число p *безопасно* тогда и только тогда, когда $(p-1)/2$ также является простым числом. Примеры безопасных простых чисел — 83, 107 и $10^{100} - 166517$. Очевидно, что генерация безопасных простых чисел p и q намного труднее, чем генерация обычных простых чисел. Открытой является следующая задача — существует бесконечно много или нет безопасных простых чисел.

Имеются другие свойства p, q и $\varphi(n)$, которые могут облегчить факторизацию и расшифрование. Разработчик криптосистемы RSA должен учитывать их, а также свойства, которые могут быть обнаружены в будущем. Действительно, наиболее очевидные среди подобных свойств учитываются в существующем для RSA компьютерном обеспечении.

Выбор p и q также важен с точки зрения возможной атаки, основанной на *последовательном расшифровании*. Это означает, что процесс начинается с криптотекста c_0 и вычисления чисел

$$c_i = (c_{i-1}^e, \mod n), \quad i = 1, 2, \dots ,$$

пока не найдут осмысленного c_i. Нетрудно показать, что возможность успеха такой атаки маловероятна, если $p-1$ и $q-1$ имеют большие простые множители p' и q', а также $p'-1$ и $q'-1$ имеют большие простые множители. Легко оценить вероятность успеха по размерам используемых простых множителей.

Пусть p и q уже выбраны. Рассмотрим выбор e и d. Он не является независимым, потому что одно из них определяет второе. Обычно d не должно быть мало, иначе оно может быть найдено перебором. Это является аргументом в пользу того, почему мы при проектировании системы сначала фиксируем d, а затем вычисляем e.

Однако маленькое e также может привести к риску для безопасности, как показано, к примеру, в [Wie]. Если одинаковое сообщение посылается нескольким получателям, то криптоанализ может стать возможным. Пусть A, B, C имеют открытую экспоненту зашифрования 3, тогда как модулями являются n_A, n_B, n_C. (Мы полагаем также, что любые два модуля взаимно просты.) Итак, передаются сообщения

$$(w^3,\ \bmod n_i), \quad i = A,\ B,\ C\ .$$

Криптоаналитик, перехватывающий эти сообщения, может вычислить число

$$w_1 = (w^3,\ \bmod n_A n_B n_C)$$

по китайской теореме об остатках. Так как w меньше каждого из модулей, мы должны иметь $w^3 = w_1$. Это означает, что криптоаналитик может найти w извлечением кубического корня из w_1.

Если $n_A = 517, n_B = 697, n_C = 667$ и тремя перехваченными сообщениями являются 131, 614 и 127, то криптоаналитик вычисляет сначала инверсии $m_i^{-1} \pmod{n_i}$, $i = A,\ B,\ C$, где m_i — произведение двух других модулей. В этом случае произведения равны 464899, 344839, 360349 и их инверсии 156, 99, 371. Следовательно,

$$w_1 \equiv 131 \cdot m_A \cdot m_A^{-1} + 614 \cdot m_B \cdot m_B^{-1} + 127 \cdot m_C \cdot m_C^{-1} \pmod{n_A n_B n_C}.$$

Так как $n_A n_B n_C = 240352783$, криптоаналитик заключает, что

$$w_1 = w^3 = 91125000\ ,$$

из которого получается исходный текст $w = 450$.

Хотя, с точки зрения безопасности, желательно, чтобы оба e и d являлись большими, малые e и d предпочтительнее, когда критично время выполнения зашифрования или расшифрования. Небольшие экспоненты полезны, когда существует большая разница в вычислительных мощностях между двумя связывающимися сторонами. Типичным является пример, когда RSA используется для связи пользователей кредитных карточек с большим компьютером. Тогда очень полезно для пользователей иметь небольшое d, а для большого компьютера иметь небольшое e. В ситуациях, как в этой, должен быть достигнут компромисс между безопасностью и доступными вычислительными мощностями.

Обратим внимание, наконец, на курьез, что в каждой криптосистеме RSA некоторые блоки исходного сообщения при зашифровании переходят сами в себя. Действительно, существует по меньшей мере *четыре* блока исходного текста, удовлетворяющих обоим условиям $E(w) = w$ и $(w, n) = 1$. Очевидно, что

$$(1^e, \mod p) = (1^e, \mod q) = 1 \quad \text{и}$$

$$((p-1)^e, \mod p) = p - 1, \quad ((q-1)^e, \mod q) = q - 1 \ ,$$

последние равенства следуют из того факта, что e всегда нечетно. Мы получаем по китайской теореме об остатках одновременное решение сравнений

$$x \equiv a \pmod p \qquad \text{и} \qquad x \equiv b \pmod q \ .$$

Когда мы потребуем, чтобы a и b принимали независимые значения ± 1, мы получим четыре числа w, удовлетворяющих

$$(w^e, \mod n) = w \ .$$

В первой иллюстрации примера 4.1, четыре таких числа w — 1, 21, 34, 54. В этом порядке они соответствуют парам (a, b): $(1, 1)$, $(1, -1)$, $(-1, 1)$ и $(-1, -1)$.

Если условие $(w, n) = 1$ игнорируется, то $w = 0$ также может быть исходным текстом и существует, по меньшей мере, *девять* чисел w с $E(w) = w$. Это видно точно так же, как и ранее, учитывая, что теперь возможным значением для a и b является и 0. В примере 4.1 мы получаем следующие пять дополнительных значений: 0, 45, 10, 11, 44.

Дискуссия, проведенная выше, показывает, что определенных исходных текстов нужно избегать. Также должны игнорироваться определенные экспоненты зашифрования e. Если $e - 1$ является кратным обоих чисел $p-1$ и $q-1$, то каждое w удовлетворяет равенству $E(w) = w$. Это непосредственно следует из теоремы Эйлера. Таким образом, $e = \varphi(n)/2 + 1$ является особенно плохим выбором, хотя и лежит в обычном диапазоне для e.

4.3. Проверка чисел на простоту

Этот параграф представляет некоторые основные факты, включающие проверку чисел на простоту и факторизацию, особенно с точки зрения RSA. Для более детального обсуждения читатель может обратиться к [Ko] и к ссылкам, встречающимся далее.

Задача PRIMALITY(n) уже рассматривалась в параграфе 2.2. Эффективный алгоритм для этой задачи очень важен при проектировании

RSA-криптосистем. Неизвестно, принадлежит ли эта задача классу P. Однако это несущественное препятствие с точки зрения RSA, потому что мы конструируем только простые числа определенного размера и, кроме того, вполне приемлемыми являются стохастические алгоритмы с низкой вероятностью неуспеха.

Такие стохастические алгоритмы работают в большинстве случаев, как описано далее. Рассмотрим *тест на составность* $C(m)$. Если целое число m успешно проходит тест, то оно является составным. Если m не проходит теста, m может быть простым. Вероятность для m быть простым числом возрастает с числом неудачных тестов на составность. Даже если m минует тест на составность, мы еще столкнемся с очень трудной задачей факторизации m. Как мы уже упоминали, безопасность RSA основана на допущении, что намного легче найти два больших простых числа p и q, чем раскрыть их, если известно только их произведение n. Это допущение основано лишь на эмпирических данных, доказанных теорем такого рода не существует.

Так как разработчик RSA-криптосистемы сталкивается с очень маловероятной возможностью, что сгенерированное им число p действительно является составным, исследуем, что может означать такая ошибка. Если $p = p_1p_2$, где p_1, p_2, как и q, являются простыми числами, то разработчик работает с ложной $\varphi_1(n) = (p-1)(q-1)$, тогда как правильной будет функция $\varphi(n) = (p_1-1)(p_2-1)(q-1)$. Пусть u — наименьшее общее кратное чисел p_1-1, p_2-1, $q-1$. Пусть также $(w,n) = 1$. Тогда по теореме Эйлера справедливы сравнения

$$w^{p_1-1} \equiv 1 \pmod{p_1}, \quad w^{p_2-1} \equiv 1 \pmod{p_2}, \quad w^{q-1} \equiv 1 \pmod{q}$$

и сравнение $w^u \equiv 1$ верно для всех трех модулей. Из этого следует

$$w^u \equiv 1 \pmod{n} .$$

Очевидно, u делит $\varphi(n)$. Если u делит также $\varphi_1(n)$, то

$$w^{\varphi_1(n)+1} \equiv w \pmod{n} ,$$

откуда следует, что зашифрование и расшифрование выполняются так, как если бы p было простым числом. Это случается, к примеру, если разработчик криптосистемы выбирает $p = 91$, $q = 41$. Тогда

$$n = 3731, \qquad \varphi_1(n) = 3600, \qquad \varphi(n) = 2880 .$$

Наименьшее общее кратное u чисел 6, 12 и 40 равно 120, числу, делящему $\varphi_1(n) = 3600$. Из этого условия также следует, что если $(d, \varphi_1(n)) = 1$, то также $(d, \varphi(n)) = 1$. Итак, по ложной функции φ_1

можно вычислить e, для которого снова будет выполняться равенство $D(E(w)) = w$. Это никак не повлияет на безопасность системы кроме того, что наименьшее общее кратное будет значительно меньше $\varphi(n)$.

Однако если u не делит $\varphi_1(u)$, то в большинстве случаев $D(E(w)) \neq w$, и этот факт, вероятно, будет замечен разработчиком. Пусть в качестве простых чисел выбраны $p = 391$ и $q = 281$, хотя $391 = 17 \cdot 23$. Тогда проектировщик системы работает с

$$n = 109871 \qquad \text{и} \qquad \varphi_1(n) = 109200 \ ,$$

тогда как в действительности $\varphi(n) = 98560$. В этом случае $u = 6160$. В самом деле, каждое из чисел 16, 22, 280 делит 6160. Однако u не делит $\varphi_1(n)$, это следует из того факта, что 11 делит u, но не делит $\varphi_1(n)$. Разработчик может теперь выбрать $d = 45979$ и вычислить $e = 19$. Для исходного текста $w = 8$ получим

$$w^{ed-1} = 8^{873600} \equiv 66879 \pmod{109871} \ .$$

Для вычисления этого отметим, что $8^{11} \equiv 70 \pmod{n}$, и учитываем, что $8^{6160} \equiv 1 \pmod{n}$.

Пусть m — нечетное целое число и $(w, m) = 1$. Если m простое, то по теореме Эйлера (также называемой в этом случае малой теоремой Ферма)

$$(*) \qquad w^{m-1} \equiv 1 \pmod{m} \ .$$

Если m не является простым, выполнение (*) возможно, но маловероятно. В этом случае m является *псевдопростым по основанию* w. Это немедленно дает следующий тест на составность: m успешно проходит тест $C(m)$ тогда и только тогда, когда

$$(*)' \qquad w^{m-1} \neq 1 \pmod{m} \ ,$$

для некоторого w_i с $(w, m) = 1$. Если m не удовлетворяет тесту $C(m)$ для w, т.е. выполняется (*), то m еще может быть составным.

Возьмем число $m = 91$, рассмотренное выше. Тогда $2^{90} = 2^{64}2^{16}2^8 2^2 \equiv 16 \cdot 16 \cdot 74 \cdot 4 \equiv 64 \pmod{91}$, что говорит о том, что 91 — составное. С другой стороны, $3^{90} \equiv 1 \pmod{91}$, что говорит о том, что 91 — псевдопростое по основанию 3. Можно аналогично доказать, что 341 — псевдопростое по основанию 2 и 217 — псевдопростое по основанию 5. В действительности можно достаточно легко показать, что три числа 341, 91 и 217 являются наименьшими псевдопростыми числами по основаниям 2, 3 и 5 соответственно.

Назовем целое число w, для которого $(w, m) = 1$ и выполнено сравнение (*), *свидетелем* простоты числа m. Как мы видим, существуют также и "ложные свидетели", для которых m является только псевдопростым. Метод, с высокой вероятностью показывающий, что m простое, состоит из отбора большого числа свидетелей простоты m. Следующий результат обеспечивает некоторое теоретическое обоснование этого факта.

Лемма 4.2 *Все или не более половины целых чисел w, где $1 \leq w < m$ и $(w, m) = 1$, являются свидетелями простоты m.*

Доказательство. Пусть w не является свидетелем (выполняется $(*)'$). Пусть w_i, $1 \leq i \leq t$, — это все свидетели. Тогда числа

$$u_i = (ww_i, \mod m), 1 \leq i \leq t,$$

попарно различны и удовлетворяют условиям $1 \leq u_i < m$ и $(u_i, m) = 1$. Нет числа u_i, которое может быть свидетелем, потому что

$$1 \equiv u_i^{m-1} \equiv w^{m-1} w_i^{m-1} \equiv w^{m-1} \pmod{m}$$

будет противоречить $(*)'$. Существует столько же чисел u_i, сколько всех свидетелей.

□

Вероятностный алгоритм работает следующим образом. Задав m, выбираем случайное w, где $1 \leq w < m$. Наибольший общий делитель (w, m) находится с помощью алгоритма Евклида. Если $(w, m) > 1$, заключаем, что m составное. В противном случае вычисляем $u = (w^{m-1}, \mod m)$ последовательными возведениями в квадрат. Если $u \neq 1$, заключаем, что m составное. Если $u = 1$, то w — свидетель простоты m и мы имеем некоторое обоснование, что m может быть простым. Чем больше свидетелей мы найдем, тем сильнее будет это обоснование. Когда мы найдем k свидетелей, по лемме 4.2 вероятность того, что m будет составным, не превосходит 2^{-k}, так как в худшем случае все числа w (с $(w, m) = 1$ и $w < m$) являются свидетелями. Если m — простое, то все числа являются свидетелями, и обоснование приводит к правильному заключению. Однако все числа могут быть свидетелями для m, *не являющегося* простым. Такие числа m называются *числами Кармишеля*. По определению, нечетное составное число m называется *числом Кармишеля* тогда и только тогда, когда выполняется (*) для всех w с $(w, m) = 1$.

Легко доказать, что число Кармишеля никогда не является квадратом другого числа и что нечетное составное число m, не являющееся квадратом, есть число Кармишеля тогда и только тогда, когда для простого p, делящего m, $p-1$ делит $m-1$. Из этого следует, что число Кармишеля должно быть произведением не менее трех различных простых чисел. К примеру, для $m = 561 = 3 \cdot 11 \cdot 17$ каждое из трех чисел $3-1$, $11-1$, $17-1$ делит $561-1$ и, следовательно, 561 является числом Кармишеля. В действительности это наименьшее число Кармишеля.

Числами Кармишеля являются также 1729, 294409 и 56052361. Все они имеют вид $(6i+1)(12i+1)(18i+1)$, где три множителя являются простыми числами. (Указанные выше числа получаются для значений $i = 1, 6, 35$.) Все числа этого вида, где три множителя являются простыми числами, являются числами Кармишеля. Существуют также числа Кармишеля другого вида, к примеру 2465 и 172081. Неизвестно, бесконечно ли множество чисел Кармишеля. Оценка вероятности 2^{-k} для алгоритма, описанного выше, не верна, если проверяемое число m является числом Кармишеля. С помощью этого алгоритма мы только имеем шанс найти, что m составное, попав при случайном выборе на число w с $(w, m) > 1$.

Теперь опишем тест, называемый *тестом проверки простоты Соловея–Штрассена*. Он очень похож на тест, описанный выше, за исключением того, что вместо условия (*) используется другое условие (**). Однако для последнего условия нет аналогов чисел Кармишеля. Таким образом, чем больше свидетелей мы найдем, тем выше будет вероятность того, что проверяемое число является простым. В приложении В читатель может найти определение символов Лежандра и Якоби $\left(\frac{w}{m}\right)$.

Лемма 4.3 *Если m — простое, то для всех w*

$$(**) \qquad w^{(m-1)/2} \equiv \left(\frac{w}{m}\right) \pmod{m} .$$

Доказательство. Очевидно, (**) выполняется, если m делит w. Иначе, по малой теореме Ферма,

$$(w^{m-1}, \bmod m) = 1, \text{ дающее } (w^{(m-1)/2}, \bmod m) = \pm 1 .$$

Пусть g — образующая $F^*(m)$ (см. параграф 3.5) и $w = g^j$. Тогда $\left(\frac{w}{m}\right) = 1 \Leftrightarrow j$ — четно $\Leftrightarrow (w^{(m-1)/2}, \bmod m) = 1$. Итак, обе части (**) сравнимы с ± 1 и сравнимы с $+1$ тогда и только тогда, когда j — четно.

□

Нечетные составные m, удовлетворяющие (**), для некоторого w с $(w,m)=1$, называются *псевдопростыми эйлеровыми числами* по основанию w. Так как (**) влечет (*), то эйлеровы псевдопростые числа по основанию w также являются и псевдопростыми числами по основанию w. Обратное неверно: 91 — псевдопростое, но не является эйлеровым псевдопростым по основанию 3, так как выполняется (*), но $3^{45} \equiv 27 \pmod{91}$, откуда следует, что (**) не выполняется. 91 есть эйлеровое псевдопростое число по основанию 10. Следующая лемма является аналогом леммы 4.2, но связана с (**) вместо (*).

Лемма 4.4 *Если m — нечетное составное число, то не более половины целых чисел w, где $(w,m)=1$ и $1 \le w < m$, удовлетворяет условию* (**).

Доказательство. Вначале построим w', такое, что (**) не выполняется (для $w=w'$). Пусть квадрат p^2 простого числа p делит m. Тогда мы выбираем $w' = 1 + m/p$. Теперь $\left(\frac{w'}{m}\right) = 1$, но левая часть (**) не сравнима с 1 $(\mathrm{mod}\ m)$, так как p не делит $(m-1)/2$.

Пусть m — произведение различных простых чисел и p одно из них. Выберем любой квадратичный невычет s по модулю p и пусть w', где $1 \le w' < n$, удовлетворяет сравнениям

$$w' \equiv s \pmod{p}, \quad w' \equiv 1 \pmod{m/p}\ .$$

Такое w' может быть найдено с помощью китайской теоремы об остатках. Тогда $\left(\frac{w'}{m}\right) = -1$, но

$$(w')^{(m-1)/2} \equiv 1 \pmod{m/p},\ \text{дающее}\ (w')^{(m-1)/2} \not\equiv -1 \pmod{m}\ .$$

Получив w', для которого условие (**) не выполняется, рассмотрим все целые числа w_i, $1 \le i \le t$, удовлетворяющие (**) (как обычно с условиями $1 \le w_i < t$, $(w_i,m)=1$). Все числа

$$u_i = (w' w_i,\ \mathrm{mod}\, m),\ 1 \le i \le t$$

различны и удовлетворяют условиям $1 \le u_i < m$ и $(u_i,m)=1$. Если некоторое u_i удовлетворяет (**), получим

$$(w')^{(m-1)/2} w_i^{(m-1)/2} \equiv \left(\frac{w'}{m}\right)\left(\frac{w_i}{m}\right) \pmod{m}\ .$$

Так как w_i удовлетворяет (**), то далее имеем

$$(w')^{(m-1)/2} \equiv \left(\frac{w'}{m}\right) \pmod{m}\ ,$$

противоречащее факту, что w' не удовлетворяет (**). Следовательно, ни одно из чисел u_i не удовлетворяет (**). Их будет столько же, сколько чисел w_i.

□

Тест Соловея–Штрассена использует условие (**) точно в том же смысле, что и наш прежний алгоритм использовал (*). Проверяя простоту m, мы вначале выбираем случайное число $w < m$. Если $(w, m) > 1$, то m — составное. Иначе проверяем выполнимость (**). Это легко, с точки зрения сложности, так как значение $\left(\frac{w}{m}\right)$ может быть вычислено быстро с помощью закона квадратичной обратимости. Если (**) не выполнено, то m — составное. Иначе w является свидетелем простоты m, выбираем другое случайное число меньшее m и повторяем процедуру. После нахождения k свидетелей заключаем с помощью лемм 4.3 и 4.4, что вероятность того, что m составное, не превосходит 2^{-k}. Результат получается сильнее, чем с помощью предыдущего алгоритма, так как лемма 4.4 показывает, что не существует аналогов чисел Кармишеля при работе с (**). Однако оценка леммы 4.4 не может быть улучшена. Существуют числа m, которые являются эйлеровыми псевдопростыми точно на половине всех возможных оснований. Примерами являются ранее упомянутые числа Кармишеля 1729 и 56052361.

Существует и другая модификация теста проверки простоты, где оценка действительно может быть улучшена: не более 25% возможных чисел являются (ложными) свидетелями простоты для составного числа m. Опишем этот тест, известный как *тест Миллера–Рабина*. Некоторые теоретико-числовые факты будут представлены без доказательств (для подробного ознакомления см. [Ко]). Доказательства являются более сложными, чем для предыдущих тестов.

Пусть m псевдопростое по основанию w, т.е. выполнено (*). Идея состоит в последовательном извлечении квадратных корней из сравнения (*) и проверки, что первое отличное от 1 число в правой части сравнения действительно равно -1. Если m простое, первое такое число должно равняться -1, потому что тогда только ± 1 являются квадратичными корнями по модулю m. Итак, мы получили другой тест проверки на простоту. Если m не удовлетворяет этому тесту, т.е. первое число, отличное от 1, равно -1, но m составное, то m называется *сильным псевдопростым* числом по основанию w.

Представим тест более подробно. Пусть m — нечетное составное число и s — максимальная степень двойки, которая делит $m - 1$, т.е.

$$m - 1 = 2^s r, \text{ где } r \text{ — нечетно .}$$

Выберем число w с $1 \leq w < m$ и $(w, m) = 1$. Значит, m *сильное псевдопростое* по основанию w тогда и только тогда, когда выполняются следующие условия:

(***) $\qquad$ или $w^r \equiv 1 \pmod{m}$ или $w^{2^{s'}r} \equiv -1 \pmod{m}$,

для некоторого s' с $0 \leq s' < s$.

Заметим, что формальное определение уточняет идею извлечения квадратных корней из сравнения

$$w^{m-1} = w^{2^s r} \equiv 1 \pmod{m} .$$

Нет квадратных корней, которые могут быть извлечены, если в левой части останется w^r.

Можно показать, что сильное псевдопростое число m по основанию w является также эйлеровым псевдопростым числом по основанию w. Если $m \equiv 3 \pmod{4}$, то также верно и обратное: в этом случае эйлеровое псевдопростое число m по основанию w является также сильным псевдопростым числом по основанию w.

В тесте Миллера–Рабина мы по заданному нечетному целому числу m сначала вычисляем $m-1 = 2^s r$, где r нечетно. Как и ранее выбираем случайное число w и проверяем (***). Если тест не выполняется, то m — составное. Иначе w является свидетелем простоты m (в этом случае m — простое или сильное псевдопростое число по основанию w), и мы повторяем процедуру для другого w. После нахождения k свидетелей простоты m можно заключить, что вероятность того, что m составное, не превосходит 4^{-k}. Это является результатом следующей леммы.

Лемма 4.5 *Если m нечетное составное целое число, то m является сильным псевдопростым числом по основанию w для не более 25% от всех w, удовлетворяющих неравенствам $1 \leq w < m$.*

Чтобы быть почти уверенным в том, что m простое, совсем не обязательно проверять большое число оснований w, если m является сильным псевдопростым числом по каждому из этих оснований. К примеру, рассмотрим четыре основания 2, 3, 5, 7. Только одно составное число $m < 2.5 \cdot 10^{10}$, а именно $m = 3215031751$, является сильным псевдопростым по каждому из этих четырех оснований.

Можно сделать даже более общее утверждение, полагая, что верна “обобщенная гипотеза Римана”. При этом предположении, если m нечетное составное целое число, то (***) нарушается, по крайней мере для одного $w < 2(\ln m)^2$. Таким образом, достаточно проверить числа w только до этой границы. На этом пути тест Миллера–Рабина

трансформируется в детерминированный алгоритм с полиномиальным временем работы. (Обычная гипотеза Римана состоит в утверждении, что все комплексные нули Римановой дзета-функции, которые лежат на "критической полосе" (где действительная часть меняется от 0 до 1), на самом деле лежат на "критической линии" (где действительная часть равна 1/2). Обобщенная гипотеза Римана состоит из того же самого утверждения для обобщений дзета-функций, называемых L-сериями Дирихле.)

Пусть n является модулем RSA. Если мы можем найти w такое, что n псевдопростое, но не сильное псевдопростое по основанию w, то мы в состоянии факторизовать n. Это верно, так как в этом случае можно найти число $u \not\equiv \pm 1 \pmod{n}$ такое, что $u^2 \equiv 1 \pmod{n}$, откуда следует, что $(u+1, n)$ есть нетривиальный множитель n. Можно избежать этого при разработке криптосистемы, если быть уверенным, что $p-1$ и $q-1$ не имеют большого общего делителя.

Мы еще вернемся к вопросам, связанным с (***), в параграфе 4.4.

Только старейший и самый медленный тест, *решето Эратосфена*, действительно дает простой множитель m, откуда следует, что m составное. Решето состоит из проверок делимости m на простые числа $\leq \sqrt{m}$. Все более быстрые тесты проверки чисел на простоту обычно говорят только о том, что m составное, не упоминая о множителях.

Известно много методов факторизации. Мы не обсуждаем их здесь, так как ни один из них не осуществим для стандартной системы RSA с n, состоящим приблизительно из 200 разрядов. Асимптотически самые быстрые алгоритмы факторизации требуют времени работы

$$O(e^{\alpha\sqrt{\ln n \ln\ln n}}) \ ,$$

где константа $\alpha = 1+\epsilon$ для произвольно малого ϵ. Ко времени написания этой книги факторизация 100-разрядных чисел была еще практически неосуществимой.

4.4. Криптоанализ и факторизация

Как мы уже отмечали, нет доказательств того, что для криптоанализа RSA в действительности необходима факторизация n. Предположим, что криптоанализ может быть осуществлен другими способами. Однако если эти способы приоткрывают некоторые части секретной лазейки, то они также приводят и к быстрой факторизации n. Покажем это. Первый результат очень прост.

Лемма 4.6 *Любой алгоритм для вычисления $\varphi(n)$ применим для факторизации n без увеличения сложности.*

Доказательство. Множитель p может быть сразу же вычислен из уравнений $p+q = n - \varphi(n) + 1$ и $p - q = \sqrt{(p+q)^2 - 4n}$.

□

Пусть теперь мы имеем метод для вычисления экспоненты дешифровки d. Мы хотим показать, как этот метод может быть использован для факторизации n. Случай не так прост, как в лемме 4.6. Более того, результирующий алгоритм для факторизации будет вероятностным. Вероятность неудачи может быть сделана сколь угодно малой. Сложность нового алгоритма существенно не превышает сложности алгоритма для вычисления d. Конечно, она зависит от фиксированной вероятности, но для любой вероятности новый алгоритм вычисляет d за полиномиальное время.

Теорема 4.1 *Алгоритм для вычисления d может быть преобразован в вероятностный алгоритм для факторизации n.*

Доказательство. Доказательство основано на тех же идеях, что и при обсуждении псевдопростых и сильных псевдопростых чисел в параграфе 4.3. Представим доказательство, не зависящее от последнего обсуждения, по двум причинам. Во-первых, вместо произвольного m мы связаны здесь со специальным случаем модулей n RSA, и, во-вторых, читатель может изучить теорему 4.1 без прочтения предыдущего параграфа.

В доказательстве используются числа w, удовлетворяющие условиям $1 \leq w < n$ и $(w, n) = 1$. Эти условия далее не повторяются, хотя имеются в виду. Очевидно, что если случайный выбор $w < n$ удовлетворяет неравенству $(w, n) > 1$, то мы тут же получим множитель n. Это будет верным и в том случае, если мы найдем нетривиальный квадратный корень из $1 \pmod n$, т. е. число u со свойствами:

$$u \not\equiv \pm 1 \pmod n \text{ и } u^2 \equiv 1 \pmod n .$$

Тогда $(u+1)(u-1)$ делится на n, но не является множителем, и, следовательно, $(u+1, n)$ равно p или q. (Это уже отмечалось в параграфе 4.3.)

Так как данный алгоритм вычисляет d, мы можем немедленно получить $ed - 1$ в виде

$$ed - 1 = 2^s r, \quad s \geq 1, \quad r \text{ — нечетно} .$$

Так как $ed-1$ является кратным $\varphi(n)$, мы получаем сравнение

$$w^{2^s r} \equiv 1 \pmod{n}$$

для произвольного w. (Вспомним дополнительные условия для w.) Следовательно, для некоторого s', где $0 \le s' \le s$, s' является наименьшим числом, для которого выполнено сравнение

$$w^{2^{s'} r} \equiv 1 \pmod{n} \ .$$

Если теперь

$$(*) \qquad s' > 0 \text{ и } w^{2^{s'-1} r} \not\equiv -1 \pmod{n} \ ,$$

мы найдем нетривиальный квадратный корень из $1 \pmod{n}$ и, следовательно, завершим доказательство.

Пусть (*) не выполняется, т.е.

$$(*)' \qquad w^r \equiv 1 \pmod{n} \text{ или } w^{2^t r} \equiv -1 \pmod{n} \text{ для } t,\ 0 \le t < s \ .$$

Здесь первое сравнение говорит о том, что мы в состоянии свести s' к 0, а второе, что значение $s'-1=t$ действительно приводит к некоторому сравнению с -1.

Теперь определим верхнюю границу для чисел w, удовлетворяющих $(*)'$. Такие числа w нежелательны с точки зрения факторизации, тогда как числа w, удовлетворяющие (*), предпочтительны.

Запишем $p-1$ и $q-1$ в форме

$$p-1 = 2^i a, \quad q-1 = 2^j b, \quad \text{где } a \text{ и } b \text{ — нечетны} \ .$$

Не теряя общности, считаем, что $i \le j$. Так как $2^s r$ является кратным $\varphi(n)$, то r есть кратное ab. Следовательно, если $t \ge i$, то $2^t r$ — кратное $p-1$ и

$$w^{2^t r} \equiv 1 \pmod{p} \ .$$

Далее получаем

$$w^{2^t r} \not\equiv -1 \pmod{p}, \text{ откуда } w^{2^t r} \not\equiv -1 \pmod{n} \ .$$

Это означает, что $(*)'$ никогда не выполнено для $t \ge i$. Так как $i < s$, то можно записать $(*)'$ в эквивалентной форме

$$(*)'' \qquad w^r \equiv 1 \pmod{n} \text{ или } w^{2^t r} \equiv -1 \pmod{n}, \text{ для } t,\ 0 \le t < i \ .$$

Теперь оценим количество чисел w, удовлетворяющих первому сравнению в $(*)''$. Пусть g — образующая $F^*(p)$ и $w \equiv g^u \pmod{p}$. (Отметим, что в этом доказательстве мы говорим о числах, которые мы не в состоянии вычислить. Они используются только для проверки справедливости алгоритма и не появляются при выполнении.) Очевидно, каждое из сравнений

$$w^r \equiv 1 \pmod{p} \text{ и } ur \equiv 0 \pmod{p-1}$$

влечет другое. Следовательно, сравнения имеют одинаковое число решений относительно неизвестных w и u. Число решений последнего сравнения равно $(r, p-1) = a$. Значит, a является числом решений первого сравнения.

Аналогично получаем, что b является числом решений сравнения $w^r \equiv 1 \pmod{q}$. Из этого следует, что ab является числом решений сравнения $w^r \equiv 1 \pmod{n}$. (Заметим, что каждая пара решений для p- и q- сравнений дает решение для n-сравнения по китайской теореме об остатках. Всего таких пар ab.)

Оценим теперь количество w, удовлетворяющих второму условию в $(*)''$. Аргументируя, как и ранее, мы заключаем, что число решений w для сравнения

$$w^{2^{t+1}r} \equiv 1 \pmod{p} \quad (\text{соответственно } w^{2^t r} \equiv 1 \pmod{p})$$

равно $(2^{t+1}r, p-1) = 2^{t+1}a$ (соответственно $(2^t r, p-1) = 2^t a$). Это верно, так как $t+1 \le i$. Следовательно, число решений сравнения

$$w^{2^t r} \equiv -1 \pmod{q}$$

не превосходит $2^{t+1}a - 2^t a = 2^t a$.

Аналогично, число решений сравнения

$$w^{2^t r} \equiv -1 \pmod{q}$$

не превосходит $2^t b$. (Здесь необходимо неравенство $i \le j$: $t+1 \le i \le j$.) Поэтому число решений w сравнения

$$w^{2^t r} \equiv -1 \pmod{n}$$

не превосходит $2^t a \cdot 2^t b = 2^{2t} ab$.

Теперь мы готовы получить верхнюю границу для количества нежелательных w, т.е. w, удовлетворяющих $(*)'$, или, что эквивалентно, $(*)''$. Такая верхняя оценка получается сложением числа решений для

первого и второго сравнений в $(*)''$, последнее число будет суммой по всевозможным значениям t:

$$ab + ab\sum_{t=0}^{i-1} 2^{2t} = ab\big(1 + \sum_{t=0}^{i-1} 4^t\big) = ab\big(1 + \frac{4^i - 1}{3}\big) =$$

$$= ab\big(\frac{2}{3} \cdot 2^{2i-1} + \frac{2}{3}\big) \le ab\big(\frac{2}{3} \cdot 2^{i+j-1} + \frac{2}{3}\big) =$$

$$= ab\big(2^{i+j-1} + \frac{1}{3}(2 - 2^{i+j-1})\big) \le ab \cdot 2^{i+j-1} = \varphi(n)/2 \ .$$

(Здесь используется неравенство $1 \le i \le j$.) Так как $\varphi(n)$ равно числу всевозможных w, то не более 50% всех w нежелательны. Это означает, что после проверки k чисел w вероятность ненахождения желаемого w не превосходит 2^{-k}, быстро стремящегося к 0.

□

В вышеприведенных рассуждениях мы можем рассматривать в качестве желаемых числа w с $(w, n) > 1$. Тогда число всевозможных w равно $n - 1$ и $\varphi(n)/2$ меньше, чем 50%. Однако это улучшение оценки несущественно, так как числа w с $(w, n) > 1$ являются очень редкими исключениями. Основываясь на обобщенной гипотезе Римана, можно показать, что существует очень мало желаемых w. Поэтому теорема 4.1 может быть сформулирована, например, в виде: любой детерминированный полиномиальный алгоритм для вычисления d может быть преобразован в детерминированный полиномиальный алгоритм для факторизации n.

Система RSA также применима в ситуации, когда модули, экспоненты зашифрования и расшифрования рассылаются с помощью агентства, которому доверяют все участники. Пусть агентство публикует общий для всех модуль n, экспоненты зашифрования e_A, $e_B, \ldots$ пользователей A, $B, \ldots$. Дополнительно агентство рассылает пользователям индивидуально секретные экспоненты расшифрования d_A, $d_B, \ldots$. Простые числа p и q известны только этому агентству. Уязвимость такой схемы использования RSA следует из теоремы 4.2. Метод доказательства аналогичен методу из теоремы 4.1, а результат может быть рассмотрен как пример криптоанализа без факторизации n.

Теорема 4.2 *Для вышеописанной схемы любой пользователь в состоянии детерминированно найти за квадратичное время секретную экспоненту расшифрования другого пользователя (без факторизации n).*

Доказательство. Покажем, как B может найти d_A. Для некоторого k

$$e_B d_B - 1 = k\varphi(n) \ .$$

B не знает k, но знает e_B, d_B, e_A и n. Пусть t — наибольшее число, делящее $e_B d_B - 1$ и имеющее общий с e_A нетривиальный множитель. Обозначим

$$\alpha = (e_B d_B - 1)/t, \text{ где } (\alpha, e_A) = 1 \ .$$

Мы не можем выбрать $t = (e_B d_B - 1, e_A)$, так как, к примеру, квадрат множителя e_A может делить $e_B d_B - 1$. Существует, однако, простой детерминированный алгоритм с квадратичным временем для вычисления t и α.

Действительно, обозначим

$$e_B d_B - 1 = g_0, \qquad (g_0, e_A) = h_0$$

и определим индуктивно, для $i \geq 1$

$$g_i = g_{i-1}/h_{i-1}, \qquad (g_i, e_A) = h_i \ .$$

Для $h_i = 1$ мы имеем явно $t = h_1 h_2 \ldots h_i$ и $\alpha = g_i$. Для $h_i \geq 2$ имеем $g_{i+1} \leq g_i/2$. Это означает, что $h_i = 1$ может быть найдено за линейное число шагов, на каждом из которых используется алгоритм Евклида, что дает в совокупности квадратичную оценку времени работы.

B теперь вычисляет с помощью алгоритма Евклида a и b, такие, что

$$ax + be_A = 1 \ ,$$

где b должно быть положительным. Заметим, что $\varphi(n)$ делит α, потому что $\alpha = k\varphi(n)/t$, где k/t — целое в силу $(t, \varphi(n)) = 1$. Последнее равенство верно, так как $(e_A, \varphi(n)) = 1$ и, следовательно, t является произведением чисел, ни одно из которых не имеет нетривиальных общих множителей с $\varphi(n)$. Это наблюдение приводит к сравнению

$$be_A \equiv 1 \pmod{\varphi(n)} \ ,$$

и, следовательно, b (взятое по модулю n) может быть использовано как d_A.

□

Хотя в теореме 4.2 B конструирует d_A без факторизации n, теорема 4.1 может быть теперь использована для факторизации n.

4.5. Частичная информация в RSA

Общий вопрос о частичной информации является очень важным в криптографии. Имеется ли возможность для криптоаналитика получить некоторую частичную информацию об исходном тексте, к примеру такую, как последний бит исходного текста, хотя задача получения всего исходного текста может быть трудновычислимой? Иногда такая частичная информация может быть очень важной.

Существует много результатов для RSA относительно того эффекта, что вскрытие отдельных частей так же трудно, как и для всего текста в целом. Такие результаты имеют следующую форму. Предположим, что мы имеем алгоритм для получения определенной частичной информации о RSA, такой, как последний бит исходного сообщения, соответствующего перехваченному криптотексту. Предположим, что алгоритм работает для каждого примера криптотекстов RSA. Тогда он может быть преобразован без большого увеличения вычислительной сложности в криптоаналитический алгоритм, который вскрывает систему RSA.

Это означает, что когда RSA дает такую частичную информацию, то секретность может быть нарушена. Если мы уверены, что RSA не может быть вскрыта, мы также можем быть уверены, что невозможно получить частичную информацию, которую можно использовать. Конечно, некоторую частичную информацию всегда можно легко получить. Например, если последний десятичный разряд n есть 3, то последние десятичные разряды p и q есть 1 и 3 или 7 и 9. Такая частичная информация вряд ли приоткрывает что-либо в исходном тексте.

Является ли информация о том, что определенные части так же сложны, как и весь текст в целом, признаком криптографической силы или слабости? Можно аргументировать в обоих направлениях. Если имеется уверенность в системе, то секретность частей определенно добавляет уверенности. Когда же есть сомнения, возможность вскрытия системы с помощью частичного криптоанализа делает ситуацию еще более сомнительной.

Подходящим путем представления результатов, где существование алгоритма предполагается без рассмотрения каких-либо деталей алгоритма, является использование *оракула*. Оракул дает ответ на любой вопрос, который предполагаемый алгоритм в состоянии решить, к примеру, назвать последний бит исходного текста. Конструируемый алгоритм, допустим, алгоритм для нахождения всего исходного текста, может при своей работе задавать оракулу вопросы определенного свойства. Такие вопросы могут быть заданы, так как они не влияют на сложность. Таким образом, сложность нового алгоритма

зависит только от "дополнительных" шагов и не зависит от сложности предполагаемого алгоритма. Если сложность последнего известна, то легко оценить сложность нового алгоритма, где оракул заменяется на шаги предполагаемого алгоритма. Использование оракулов изображено на рис. 4.1.

Рис. 4.1.

Начнем с простой иллюстрации, которая показывает, как алгоритм, сообщающий о том, что исходный текст меньше, чем $n/2$ или нет, может быть использован для получения дополнительной информации о x. Итак, в нашем распоряжении имеется следующий оракул O(размер) (см. рис. 4.2).

Это означает, что по заданному входу, состоящему из открытого ключа зашифрования и зашифрованной версии x, оракул O(размер) говорит, выполняется ли $x < n/2$ или нет.

Построим алгоритм A, говорящий нам, в каком из интервалов $(jn/8, (j+1)n/8)$, $0 \le j \le 7$, лежит исходный текст x. По заданному входу, состоящему из e, n и $(x^e, \mod n)$, алгоритм только вычисляет числа

$$(*) \qquad (2^e x^e, \mod n) \text{ и } (4^e x^e, \mod n)$$

и задает три вопроса оракулу. Следовательно, увеличение сложности алгоритма A за счет любого алгоритма, выполняющего работу оракула O(размер), незначительно.

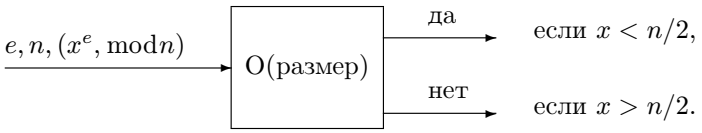


Рис. 4.2.

Один из трех вопросов, задаваемых оракулу, изображен на рис. 4.2, а в двух других x^e заменяется на $(2x)^e$ и $(4x)^e$. Последние два вопроса могут быть заданы, так как алгоритм A вычисляет числа (*). Позиция x зависит от ответов на три вопроса, согласно следующей таблице:

Ответы			Интервал
да,	да,	да	$0 < x < n/8$
да,	да,	нет	$n/8 < x < n/4$
да,	нет,	да	$n/4 < x < 3n/8$
да,	нет,	нет	$3n/8 < x < n/2$
нет,	да,	да	$n/2 < x < 5n/8$
нет,	да,	нет	$5n/8 < x < 3n/4$
нет,	нет,	да	$3n/4 < x < 7n/8$
нет,	нет,	нет	$7n/8 < x < n$

Легко проверить результаты. К примеру, пусть O(размер) дает информацию

$$x > n/2,\ (2x,\ \mathrm{mod}\, n) < n/2,\ (4x,\ \mathrm{mod}\, n) < n/2\ ,$$

т.е. последовательность ответов “нет, да, да”. Первые два неравенства говорят нам, что $n/2 < x < 3n/4$, так как если $x > 3n/4$, то $(2x,\ \mathrm{mod} n) > n/2$ и мы будем иметь второй ответ “нет”. Объединяя эту информацию с последним неравенством, мы получим $n/2 < x < 5n/8$, потому что опять $5n/8 < x < 3n/4$ влечет $(4x,\ \mathrm{mod}\, n) > n/2$.

Эта процедура может выполняться до тех пор, пока интервалы не станут такими малыми, что x будет однозначно определяться интервалом, в котором он содержится. Представим это подробнее.

Также подходящим будет использование оракула O(четность), который говорит о четности x. Если мы работаем с двоичными представлениями, то O(четность) изображается следующим образом:

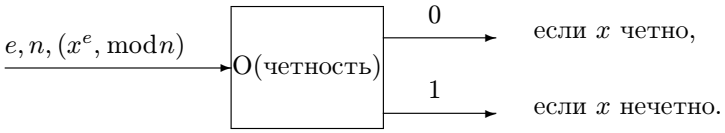


Рис. 4.3.

Таким образом, оракул сообщает последний бит x. Покажем, как, используя O(четность), можно построить x, присоединяя последовательно по одному биту справа.

Обозначим через N число битов в n. Тогда $N = [\log_2 n] + 1$. Мы также используем операторы B и M, переводящие неотрицательное число в соответствующую двоичную последовательность, и наоборот. Например, $B(91) = 1011011$ и $M(1011011) = 91$. $B(x)$ всегда начинается с 1. Операторы B и M иногда необходимы, чтобы избегать путаницы.

Для двух последовательностей битов t и u обозначим через tu последовательность битов, получаемую написанием t и u друг за другом. Последовательность tu называется *конкатенацией* t и u. Как обычно, мы обозначаем через $|t|$ *длину* последовательности t. Если $M(t) \geq M(u)$, обозначим через $\mathrm{LAST}(t - u)$ последние $|u|$ битов последовательности $B(M(t)-M(u))$, дополняемые слева нулями, если $B(M(t)-M(u)) < |u|$. В целом если $\mathrm{LAST}(t - u) = v$, то $|v| = |u|$ и для некоторого w $B(M(t) - M(u))$ есть суффикс wv. К примеру,

$$\mathrm{LAST}(1011011 - 1010111) = 0000100 \ ,$$

$$\mathrm{LAST}(1011011 - 111) = 100 \ .$$

В первом случае w пустое слово, а во втором случае $w = 1010$. Условие $M(t) \geq M(u)$ гарантирует, что $\mathrm{LAST}(t - u)$ будет всегда определено.

Пусть K будет инверсией $2^e \pmod n$, т. е.

$$2^e K \equiv 1 \pmod n \ .$$

Число K находится быстро с помощью алгоритма Евклида. Для заданного $(x^e, \mathrm{mod} n)$ мы теперь индуктивно определим $r(i)$ и $\mathrm{ANS}(i)$ для $1 \leq i \leq N$.

По определению $r(1) = (x^e, \mathrm{mod} n)$ и $\mathrm{ANS}(1)$ есть ответ оракула O(четность) на вход x^e. (Мы представляем вход в такой укороченной

форме, так как части n и e сохраняются неизменными в течение всего обсуждения.) Пусть уже определены $r(i-1)$ и $\mathrm{ANS}(i-1)$ для некоторого $i \geq 2$. Тогда

$$r(i) = \begin{cases} (r(i-1)K, \mod n), & \text{если } \mathrm{ANS}(i-1) = 0 \ , \\ ((n - r(i-1))K, \mod n), & \text{если } \mathrm{ANS}(i-1) = 1 \ , \end{cases}$$

и $\mathrm{ANS}(i-1)$ есть ответ оракула на вход $r(i-1)$. Заметим, что, по определению, $r(i)$ имеет вид $(y^e, \mod n)$ для некоторого y.

Затем индуктивно определим $t(i),\ N \geq i \geq 1$. Вначале

$$t(N) = \mathrm{ANS}(N) \ .$$

Пусть уже определено $t(i),\ i \geq 2$. Тогда

$$t(i-1) = \begin{cases} t(i)0 \ , & \text{если } \mathrm{ANS}(i-1) = 0 \ , \\ \mathrm{LAST}(B(n) - t(i)0), & \text{если } \mathrm{ANS}(i-1) = 1 \text{ и } M(t(i)0) < n \ , \\ \mathrm{LAST}(t(i)0 - B(n)), & \text{если } \mathrm{ANS}(i-1) = 1 \text{ и } M(t(i)0) > n \ . \end{cases}$$

Здесь разделение по $\mathrm{ANS}(i-1)$ на два подслучая необходимо, чтобы гарантировать определенность LAST. В действительности последний подслучай требуется тогда и только тогда, когда $i = 2$ и $M(t(2)) > n/2$. Например, $n = 21,\ B(n) = 10101$ и $t(2) = 1101$.

В качестве примера возьмем первую иллюстрацию из примера 4.1. Мы имеем $n = 55,\ e = 7,\ N = 6$ и $B(n) = 110111$. Алгоритм Евклида дает $K = 52$. Пусть $x^e = 49$. (Для упрощения записи мы пишем x^e вместо $(x^e, \mod n)$.) Получаем

$$\begin{array}{ll} r(1) = 49 \ , & \mathrm{ANS}(1) = 0 \ , \\ r(2) \equiv 49 \cdot 52 \equiv 18 \ , & \mathrm{ANS}(2) = 0 \ , \\ r(3) \equiv 18 \cdot 52 \equiv 1 \ , & \mathrm{ANS}(3) = 1 \ , \\ r(4) \equiv 54 \cdot 52 \equiv 3 \ , & \mathrm{ANS}(4) = 1 \ , \\ r(5) \equiv 52 \cdot 52 \equiv 9 \ , & \mathrm{ANS}(5) = 0 \ , \\ r(6) \equiv 9 \cdot 52 \equiv 28 \ , & \mathrm{ANS}(6) = 1 \ . \end{array}$$

Конечно, значения $\mathrm{ANS}(i)$ не вычисляются, а получаются с помощью оракула. В этом простом случае они могут быть найдены в таблице из примера 4.1.

Теперь вычислим значения $t(i)$. По определению, $t(6) = 1$ и $t(5) = 10$. Так как $\mathrm{ANS}(4) = 1$, мы получаем

$$t(4) = \mathrm{LAST}(110111 - 100) = 011 \ .$$

Аналогично,

$$t(3) = \mathrm{LAST}(110111 - 0110) = 0001 \ .$$

Оставшиеся значения опять появляются с помощью конкатенации: $t(2) = 00010$ и $t(1) = 000100$. Теперь можно проверить, что $t(1)$ есть двоичное представление x из N битов:

$$4^7 \equiv 49 \pmod{55} .$$

Это справедливо также и в общем случае.

Теорема 4.3 *В вышеопределенных обозначениях*

$$M(t(1)) = x .$$

Перед началом доказательства теоремы 4.3 заметим, что для нахождения x мы консультируемся с оракулом N раз. Дополнительно при применении алгоритма Евклида используется не более $N-1$ модульных умножений и не более $2N$ вычитаний. Таким образом, криптоаналитический алгоритм для нахождения x является очень быстрым, если с оракулом можно консультироваться без увеличения стоимости. В этом случае метод нахождения последнего бита исходного текста дает метод для нахождения всего исходного текста.

Доказательство. Для $1 \le i \le N$ мы обозначим через $u(i)$ число, удовлетворяющее сравнению

$$(u(i))^e \equiv r(i) \pmod{n}, \quad 0 < u(i) < n .$$

Такие числа $u(i)$ существуют в силу определения $r(i)$. Более подробно, соотношение $2^e r(i) \equiv \pm r(i-1) \pmod{n}$ показывает, насколько успешно могут быть построены числа $u(i)$. Обозначим также

$$v(i) = 0^j B(u(j)) ,$$

где $j = N - |B(u(i))|$. Тогда $j \ge 0$, потому что $u(i) < n$. Таким образом, $v(i)$ всегда является двоичной последовательностью длины N.

Теперь потребуем, чтобы для $N \ge i \ge 1$ существовало такое слово $w(i)$, возможно пустое, что

$$(*) \qquad v(i) = w(i)t(i) .$$

Теорема 4.3 следует из (*) при подстановке $i = 1$. Заметим, что $|t(1)| = N$, так как $|t(N)| = 1$ и длина возрастает на единицу при каждом переходе от $t(i)$ к $t(i-1)$. Так как $|v(1)| = N$, из (*) следует, что $w(1)$ должно быть пустым, а также, что $v(1)6$ и $t(1)$ — одинаковые двоичные последовательности. С другой стороны, $M(v(1)) = x$ и, следовательно, $M(t(1)) = x$.

Наше требование (*) устанавливается с помощью индукции по i. Для $i = N$ (*) справедливо, так как, по определению, последний бит $v(N)$ равен последнему биту $B(u(N))$, который, в свою очередь, равен $\mathrm{ANS}(N) = t(N)$. Пусть индуктивное предположение (*) верно для i. Рассмотрим значение $i - 1$.

Пусть вначале $\mathrm{ANS}(i-1) = 0$. Тогда

$$r(i) = (r(i-1)K, \mod n)$$

и

$$r(i-1) \equiv 2^e r(i) \equiv (2u(i))^e \pmod{n} ,$$

откуда следует, что $u(i-1) = (2u(i), \mod n)$. Если $B(u(i-1)) = B(u(i))0$, то по индуктивному предположению и определению $t(i-1)$ мы получаем

$$v(i-1) = w(i-1)t(i)0 = w(i-1)t(i-1)$$

и, следовательно, (*) выполнено для значения $i-1$, где $w(i-1)$ получается из $w(i)$ путем удаления одного начального нуля. С другой стороны, из $B(u(i-1)) \neq B(u(i))0$ следует, что $u(i-1) = 2u(i) - n$. (Очевидно, что $2u(i) < 2n$.) Поэтому $u(i-1)$ нечетно, что противоречит условию $\mathrm{ANS}(i-1) = 0$. Это показывает, что $B(u(i-1)) = B(u(i))0$.

Пусть теперь $\mathrm{ANS}(i-1) = 1$. В этом случае

$$r(i-1) \equiv -2^e r(i) \equiv -2^e (u(i))^e \equiv (-2u(i))^e \pmod{n} .$$

Здесь последнее сравнение справедливо, поскольку e нечетно. Из этого следует, что $u(i-1) = (-2u(i), \mathrm{mod} n)$. Если $n > 2u(i)$, то

$$v(i-1) = w(i-1)\mathrm{LAST}(B(n) - t(i)0) = w(i-1)t(i-1) .$$

Если $n < 2u(i)$, то

$$v(i-1) = w(i-1)\mathrm{LAST}(t(i)0 - B(n)) = w(i-1)t(i-1) .$$

Две альтернативы соответствуют разделению $\mathrm{ANS}(i-1) = 1$ на два подслучая в определении $t(i-1)$. Это завершает индуктивный шаг, и, следовательно, (*) выполнено.

□

Следующий пример 4.2 иллюстрирует различные стороны вышеуказанной конструкции.

Пример 4.2. Рассмотрим сначала, как выглядят $u(i)$ и $v(i)$ в иллюстрации, приведенной перед формулировкой теоремы 4.3. Здесь опять используется таблица из примера 4.1. Мы получаем

$$\begin{aligned} u(6) &= 7, \quad v(6) = 000111\ , \\ u(5) &= 14, \quad v(5) = 001110\ , \\ u(4) &= 27, \quad v(4) = 011011\ , \\ u(3) &= 1, \quad v(3) = 000001\ , \\ u(2) &= 2, \quad v(2) = 000010\ , \\ u(1) &= 4, \quad v(1) = 000100\ . \end{aligned}$$

Сравнивая значения $v(i)$ с ранее вычисленными значениями $t(i)$, заключаем, что $w(1)$ пусто и

$$w(2) = 0,\ w(3) = 00,\ w(4) = 011,\ w(5) = 0011,\ w(6) = 00011\ .$$

В качестве второй иллюстрации рассмотрим $n = 57$, $e = 5$, $(x^e,\ \mathrm{mod}\ n) = 48$. Вначале мы получаем $N = 6$, $B(n) = 111001$, $K = 41$, а затем следующие значения.

i	1	2	3	4	5	6
$r(i)$	48	27	24	15	12	21
ANS(i)	1	0	0	1	1	1
$t(i)$	100001	01100	0110	011	11	1
$u(i)$	33	12	6	3	27	15
$v(i)$	100001	001100	000110	000011	011011	001111

Следующая иллюстрация несколько больше. Рассмотрим $n = 8137$, $e = 517$, $(x^e,\ \mathrm{mod} n) = 5611$. В этом случае мы имеем $N = 13$, $B(n) = 1111111001001$,

$$2^{517} = 2^{512} \cdot 32 \equiv 6905 \cdot 32 \equiv 1261\ \pmod{8137}\ ,$$

откуда $K = 342$. Далее следуют значения $r(i)$, ANS(i), $t(i)$:

i	$r(i)$	ANS(i)	$t(i)$
1	5611	0	0000000010000
2	6767	0	000000001000
3	3406	0	00000000100
4	1261	0	0000000010
5	1	1	000000001
6	7795	0	11100100
7	5091	0	1110010
8	7941	1	111001
9	1936	0	01000
10	3015	0	0100
11	5868	0	010
12	5154	1	01
13	3061	0	0

Следовательно, $x = M(t(1)) = 16$. Таблица может быть заполнена быстро, если в действительности можно консультироваться с оракулом. Однако, так как мы не имеем доступа ни к одному из оракулов, значения в таблице будут вычисляться другим методом, который не может быть вычислительно реализуемым, иначе мы были бы в состоянии вскрыть RSA! В проведенных выше вычислениях было заранее известно $x = 16$. Тогда столбцы t и ANS могут быть вычислены сверху вниз. Если ANS-столбец определен, то тут же вычисляется и r-столбец. В этом частном примере мы имеем $\varphi(n) = 7956$ и $d = 277$.

□

Более сильные результаты могут быть получены для вероятностных алгоритмов. По заданному $(x^e, \mod n)$ мы всегда в состоянии угадать последний бит x с вероятностью $1/2$. Предположим тем не менее, что мы имеем при отгадывании незначительное преимущество, т.е. существует положительное δ, такое, что мы всегда в состоянии угадать последний бит x с вероятностью $1/2+\delta$. Тогда мы в состоянии вскрыть RSA. Более точно, в [SchA] показан следующий результат. Пусть оракул O(четность,δ) сообщает последний бит x с вероятностью $\geq 1/2 + \delta$ после получения на входе n, e и $(x^e, \mod n)$. Если можно консультироваться с оракулом без увеличения стоимости, то существует вероятностный алгоритм с полиномиальным временем для вычисления x по указанному входу. Алгоритм является алгоритмом типа Лас Вегаса, так как выход может быть проверен с помощью модульного возведения в степень.

Мы рассмотрели оракул, сообщающий последний бит x. Результат может быть распространен и на оракулы, информирующие о некотором

другом бите x. В частности, техника теоремы 4.3 почти без изменений применима к случаю, где оракул сообщает j-й бит с конца $B(x)$ и в конце двоичного представления n стоит не менее j единиц. В теореме 4.3 мы имеем $j = 1$.

В обсуждении, связанном с теоремой 4.3, вместо оракула О(четность) мы с тем же успехом можем использовать оракул О(размер). Действительно, для всех z, где $0 < z < n$, мы имеем $z < n/2$ тогда и только тогда, когда $(2z, \mod n)$ четно. В силу этого факта каждый из этих двух оракулов воспроизводит другой.

4.6. Дискретные логарифмы и ключевой обмен

Предположим, что в RSA публикуется только модуль n, а экспонента зашифрования e хранится в секрете. Пусть криптоаналитик перехватывает не менее одной пары (w, w^e) и пытается вскрыть систему, т.е. найти экспоненту расшифрования d при начальном условии "известен исходный текст". Тогда перед криптоаналитиком стоит задача нахождения логарифма w по основанию $w^e \pmod n$. Это специальный случай вычисления *дискретных логарифмов*.

Имеется много криптосистем, с открытым ключом и других, основанных на дискретных логарифмах. Нахождение последних представляется трудновычислимым при их использовании в качестве основы криптосистемы. Если мы рассмотрим уравнение $a^x = y$ для положительных *действительных чисел*, трудность вычисления x из a и y будет примерно той же самой, что и при нахождении y из a и x. Обе задачи сводятся к ряду умножений, делений и таблице перевода логарифмов по любому основанию. Что касается дискретных логарифмов, ситуация совершенно отлична. Модульное возведение в степень может быть осуществлено действительно быстро — мы уже обсуждали это и представили численные примеры. Вероятная трудновычислимость обратной операции, а именно взятия дискретных логарифмов, уже использовалась в параграфе 3.5.

Понятие дискретного логарифма может быть сформулировано следующим образом. Пусть g — элемент конечной группы G и y — другой элемент G. Тогда любое целое x, такое, что $g^x = y$ называется *дискретным логарифмом* y по основанию g. Очевидно, каждый элемент y группы G имеет дискретный логарифм по основанию g тогда и только тогда, когда G является циклической группой с образующей g. К примеру, в мультипликативной группе положительных целых чисел по мо-

дулю 7 только числа 1, 2, 4 имеют дискретные логарифмы по основанию 2, тогда как все числа имеют дискретные логарифмы по основанию 3 согласно таблице:

Число	1	2	3	4	5	6
Логарифм	6	2	1	4	5	3

Таблицы дискретных логарифмов в простых случаях были рассмотрены в параграфе 3.5. Конечно, группы небольшого порядка не представляют вычислительных трудностей. Существуют эффективные алгоритмы вычисления дискретных логарифмов в некоторых специальных случаях, таких как алгоритм Д.Копперсмита [Cop] для конечных полей $F(2^h)$. Однако в общем случае известные алгоритмы для вычисления дискретных логарифмов в группах порядка m приблизительно имеют одинаковую сложность относительно m, как для алгоритмов факторизации m. Если же все простые множители m малы, то очень эффективно работает алгоритм Сильвера–Полига–Хэллмана, [PoH] и [Odl]. Этот алгоритм описывается в следующем примере.

Пример 4.3. Пусть $F(q)$, $q = r^h$, является конечным полем. Рассмотрим дискретные логарифмы по основанию g, где g — образующая для $F^*(q)$.

Для каждого простого делителя p числа $q-1$ вычислим

$$a(i,p) = (g^{i(q-1)/p},\ \bmod q), \quad 0 \le i < p\ .$$

Если каждое p, делящее $q-1$, мало, то таблица, содержащая вспомогательные числа $a(i,p)$, имеет приемлемый размер.

К примеру, рассмотрим $F(181)$ и $g = 2$ (2 действительно является образующей). Тогда $180 = 2^2 \cdot 3^2 \cdot 5$ и таблица чисел $a(i,p)$ выглядит следующим образом:

p / i	2	3	5
0	1	1	1
1	180	48	59
2		132	42
3			125
4			135

Теперь вычислим дискретный логарифм z числа 62 по основанию 2. В целом если $q-1 = \prod p^\alpha$, то для нахождения дискретного логарифма x числа y по основанию g достаточно найти $(x,\ \bmod p^\alpha)$ для каждого p

в разложении $q-1$ на простые множители. Затем с помощью китайской теоремы об остатках легко вычислить x из значений $(x,\ \bmod p^\alpha)$.

Вычисляя $(x,\ \bmod p^\alpha)$, мы рассмотрим представление этого числа по основанию p:

$$(x,\ \bmod p^\alpha) = x_0 + x_1 p + \ldots + x_{\alpha-1}p^{\alpha-1}\ , \quad 0 \le x_i \le p-1\ .$$

В примере мы рассмотрим множитель $p^\alpha = 3^2$ и запишем $(z,\ \bmod 181) = x_0 + 3x_1$. Для нахождения x_0 вычислим число $(y^{(q-1)/p},\ \bmod q)$, которое равно $a(i,p)$ для некоторого i. Выберем $x_0 = i$. В примере $(62^{60},\ \bmod 181) = 48$ и, следовательно, $x_0 = 1$. Это работает и в общем случае, потому что $(g^{q-1},\ \bmod q) = 1$ и, значит,

$$y^{(q-1)/p} \equiv g^{x(q-1)/p} \equiv g^{x_0(q-1)/p} \equiv a(x_0,p)\ \pmod q\ .$$

Для получения x_1 вычислим сначала инверсию g^{-x_0} числа $g^{x_0}\ \pmod q$ и рассмотрим $y_1 = yg^{-x_0}$. Если теперь

$$(y_1^{(q-1)/p^2},\ \bmod q) = a(i,p)\ ,$$

то $x_1 = i$. Для получения x_2 мы рассмотрим число $y_2 = yg^{-x_0-x_1p}$ и вычислим

$$(y_2^{(q-1)/p^3},\ \bmod q)\ .$$

Процедура выполняется до тех пор, пока не будет найдено $(x,\ \bmod p^\alpha)$.

Возвращаясь к примеру, мы находим $y_1 = 31$. Из этого следует, что

$$(y_1^{180/9},\ \bmod 181) = 1\ ,$$

поэтому $x_1 = 0$. Вместе это дает $z \equiv 1\ \pmod 9$.

Рассмотрим следующий множитель $p^\alpha = 2^2$. Теперь мы определяем $x_0 + 2x_1$. (Мы используем те же x-обозначения для неизвестных.) Так как $(62^{90},\ \bmod 181) = 1$, мы заключаем, что $x_0 = 0$. Теперь $y_1 = y = 62$ и $(62^{45},\ \bmod 181) = 1$, откуда $x_1 = 0$ и $z \equiv 0\ \pmod 4$.

Рассмотрим, наконец, множитель $p^\alpha = 5^1$. Теперь определяем только x_0. Так как $(62^{36},\ \bmod 181) = 1$, заключаем, что $x_0 = 0$ и $z \equiv 0\ \pmod 5$. Три сравнения для z теперь дают значение $z = 100$. Следовательно, $\log 62 = 100$.

Та же таблица может быть использована для вычисления дискретного логарифма любого y, кроме значения $y = 62$. Рассмотрим $y = 30$. Обозначим $\log 30 = z$. Для множителя 2^2 мы получаем $(30^{90},\ \bmod 181) = 180$ и, следовательно, $x_0 = 1$. Так как $(15^{45},\ \bmod 181) = 132$, мы получаем, что $x_1 = 0$ и, значит, $z \equiv 1\ \pmod 4$. Для множителя 3^2 $x_0 = 0$, так как $(30^{60},\ \bmod 181) = 1$. Поскольку $(30^{20},\ \bmod 181) = 132$,

мы заключаем, что $x_1 = 2$ и $z \equiv 6 \pmod 9$. Наконец, для множителя 5 получаем, что $x_0 = 3$ и $z \equiv 3 \pmod 5$, так как $(30^{36}, \mod 181) = 125$. Три сравнения дают результат $\log 30 = 33$.

Алгоритм Сильвера–Полига–Хэллмана всегда эффективен с единственным исключением, что построение таблицы для чисел $a(i,p)$ может стать трудновычислимым, когда $q-1$ имеет большой простой множитель p. Вырожденным случаем является $F(q)$, где q — безопасное простое число (см. параграф 4.2). Вычисление $(q-1)/2$ элементов столбца для $(q-1)/2$ сумм приводит практически к той же вычислительной сложности, что и при построении таблицы логарифмов.

□

В целом криптосистемы с открытым ключом используются реже, чем классические криптосистемы. Задачи управления ключом в последних могут быть решены с помощью подходящего протокола для *ключевого обмена*. Два пользователя договариваются о секретном ключе, который позже будет использоваться в качестве основы классической криптосистемы, такой как DES или системы Плейфейра. С помощью подходящего кодирования ключ всегда можно представить в виде числа. Самые первые и также наиболее широко используемые протоколы обмена ключами полагаются на трудновычислимость задачи вычисления дискретных логарифмов. Представим кратко один из таких протоколов.

Простое число q и образующая g группы $F^*(q)$ распространяются среди всех пользователей. Каждый пользователь A_i случайно выбирает число k_i. Пользователи хранят числа k_i в секрете, а публикуют степени $(g^{k_i}, \mod q)$. Таким образом, открытая информация образует следующую таблицу.

Пользователь	A_1	A_2	$\ldots$	A_n
Число	$(g^{k_1}, \mod q)$	$(g^{k_2}, \mod q)$	$\ldots$	$(g^{k_n}, \mod q)$

Общий ключ для двух пользователей A_i и A_j, которые не имели предшествующей связи, является теперь числом $(g^{k_i k_j}, \mod q)$. Пользователь A_i может вычислить это число, используя k_i и информацию, опубликованную A_j. Ситуация симметрична с точки зрения A_j. Ключ может быть вычислен также и криптоаналитиком, который в состоянии посчитать дискретные логарифмы и, таким образом, найти k_i или k_j из открытой информации. Активный перехватчик A_m, который в состоянии вставить в таблицу число $(g^{k_m}, \mod q)$ в столбец для A_j,

способен установить связь с A_i, который на самом деле хочет связаться с A_j.

В качестве иллюстрации выберем $q = 181$ и $g = 2$ (см. пример 4.3). Пусть A_1 (соответственно A_2) выбирает $k_1 = 100$ (соответственно $k_2 = 33$). Следовательно, публикуемые числа равны -62 и 30. Теперь и A_1, и A_2 в состоянии вычислить общий ключ 48:

$$(30^{100},\ \mathrm{mod}\ 181) = (62^{33},\ \mathrm{mod}\ 181) = 48.$$

Описанная система ключевого обмена взята у Диффи и Хэллмана [DH]. Она является старейшей из систем, предлагаемых для исключения перемещений секретных ключей, а также одной из наиболее безопасных схем с открытым ключом. Она проверена с различных сторон и является практической с вычислительной точки зрения. Если q — простое число, состоящее из 1000 бит, то пользователю A_i нужно произвести примерно только 2000 умножений для вычисления ключа $(g^{k_i k_j},\ \mathrm{mod}\, q)$. С другой стороны, перехватчик вычисляет дискретные логарифмы. Это требует более 2^{100} операций при использовании любого из известных на сегодняшний день алгоритмов.

В следующей простой модификации схемы Диффи–Хэллмана сообщения передаются направленно. Пусть q — простое или степень простого числа, известное всем пользователям. Каждый пользователь A выбирает секретное целое число e_A, такое, что $0 < e_A < q - 1$ и $(e_A, q - 1) = 1$. Затем A вычисляет инверсию d_A числа $e_A\ (\mathrm{mod}\ q - 1)$. Передача сообщения w, $0 < w < q - 1$, выполняется за три следующих шага.

Шаг 1: A посылает B $(w^{e_A},\ \mathrm{mod}\, q)$.
Шаг 2: B посылает A $(w^{e_A e_B},\ \mathrm{mod}\, q)$.
Шаг 3: A применяет d_A и посылает B $(w^{e_B},\ \mathrm{mod}\, q)$.

Этот протокол является одним из основных в криптографии с открытым ключом. Мы уже встречались с несколькими его версиями. Он уязвим против активного перехватчика, который принимает сообщение на шаге 1 и маскируется под B.

В следующей модификации, Эль Гамаля [ElG], q и образующая g группы $F^*(q)$ известны всем пользователям. Каждый пользователь A выбирает секретное целое число m_A, $0 < m_A < q - 1$, и публикует g^{m_A} (рассматриваемый как элемент $F(q)$) в качестве ключа зашифрования. Сообщения w посылаются к A в форме $(g^k,\ wg^{km_A})$, где k — случайное целое число, выбранное отправителем. Для отправителя достаточно знать g^{m_A}, в то время как A может открыть w, сначала вычислив g^{-km_A}. Перехватчик же решает задачу нахождения дискретных логарифмов.

Представим, наконец, общий способ для ключевого обмена. В целом вычислительная сложность для легальных пользователей равна $O(m)$, а для перехватчиков — $O(m^2)$.

Пространство ключей размерности m^2, так же как и односторонняя функция f, известны всем пользователям. Обратная функция не известна ни одному из них. Пользователь A (соответственно B) случайно выбирает m ключей $x_1, \ldots, x_m$ (соответственно $y_1, \ldots, y_m$) и вычисляет значения $f(x_1), \ldots, f(x_m)$ (соответственно $f(y_1), \ldots, f(y_m)$). B посылает A значения $f(y_i)$ и A отвечает посылкой к B значения $f(x_j)$, которое лежит среди значений, полученных от B. При маловероятном событии, когда не существует такого $f(x_j)$, A вычисляет дальнейшие значения $f(x)$ выбором новых ключей x, пока соответствие не будет найдено. Для извлечения выгоды из данной ситуации перехватчик обычно вычисляет $f(x)$ примерно для m^2 значений x.

Глава 5

Другие основы криптосистем

5.1. Возведение в степень в квадратичных полях

Представленные в параграфе 2.2 соображения относительно построения систем с открытым ключом являются очень общими. Кроме того, никак не определялись область или предмет задачи, лежащей в основе системы. Стоит испытать любую “одностороннюю улицу” и в действительности окажется, что опробовано много вариантов. К настоящему времени существует множество криптосистем с открытым ключом, основанных на совершенно разных идеях.

Цель этой главы — дать идею построения различных типов существующих криптосистем с открытым ключом. Изложение не будет исчерпывающим, мы обсудим только несколько систем. Причем мы не делаем попытки выбрать “лучшие” системы, а только хотим представить материал, который мог бы вдохновить исследователей на работу в области создания криптосистем, не зависящих от сложности некоторых задач из теории чисел.

Как мы уже отмечали, очень редко можно получить математические результаты относительно качества криптосистем; любые факты, касающиеся безопасности системы, обычно основаны на практике. Поэтому в большинстве случаев сравнение различных систем бесполезно, так же как и выбор среди них лучших. В литературе безопасность многих криптосистем обосновывается с помощью простого утверждения,

что криптоанализ имеет такую же сложность, что и для трудновычислимой задачи. Этого не достаточно, так как данная задача может оказаться легче, чем ожидалось. В целом также не известно, действительно ли для криптоанализа необходимо решение базисной задачи; возможно, что может быть найден искусный обход.

В RSA базисной задачей является разложение на множители произведения двух простых чисел. Как мы уже отмечали, сложность этой задачи неизвестна. Также неизвестно, существуют ли обходные пути для криптоанализа. Следующая криптосистема, разработанная Уильямсом [Wil], кажется, обладает всеми достоинствами RSA. Кроме того, можно в действительности доказать, что любой метод вскрытия этой системы с помощью предварительного криптоанализа приводит к факторизации модулей. Таким образом, предварительный криптоанализ эквивалентен факторизации. С другой стороны, всегда успешен криптоанализ при начальном условии "известен избранный криптотекст". Поэтому при использовании этой системы данное начальное условие должно быть невозможным для перехватчика.

Для системы Уильямса зашифрование и расшифрование сравнимы по скорости с аналогичными операциями системы RSA. Однако описание этой системы намного труднее. Вместо обычного модульного возведения в степень, такого, как в RSA, здесь возводятся в степень по модулю n числа вида $a + b\sqrt{c}$. При этом числа a, b и c являются целыми, но квадратный корень может быть иррациональным. Тем не менее нет необходимости в вычислении квадратных корней. Далее все детали будут описаны без использования терминологии квадратичных полей.

Рассмотрим числа вида

$$\alpha = a + b\sqrt{c}\ ,$$

где a, b и c — целые. Здесь $\sqrt{c}$ формально понимается как число, квадрат которого равен c. Если зафиксировать c, то числа α могут рассматриваться как пары (a, b), для которых операции сложения и умножения определены следующим образом:

$$\alpha_1 + \alpha_2 = (a_1, b_1) + (a_2, b_2) = (a_1 + a_2, b_1 + b_2)\ ,$$

$$(a_1, b_1) \cdot (a_2, b_2) = (a_1 a_2 + c b_1 b_2, a_1 b_2 + b_1 a_2)\ .$$

Сопряженным с α назовем число

$$\overline{\alpha} = a - b\sqrt{c}\ .$$

В дальнейшем будут широко использоваться функции X_i и Y_i, где $i = 0, 1, 2, \ldots$. По определению

$$X_i(\alpha) = X_i(a, b) = (\alpha^i + \overline{\alpha}^i)/2 \ ,$$

$$Y_i(\alpha) = Y_i(a, b) = b(\alpha^i - \overline{\alpha}^i)/(\alpha - \overline{\alpha}) = (\alpha^i - \overline{\alpha}^i)/2\sqrt{c} \ .$$

(Последнее выражение предназначено только для упрощения понимания. Так как функции определены для пар (a, b), то не содержат c.) Тогда степени α и $\overline{\alpha}$ могут быть выражены через функции X_i и Y_i:

$$\begin{aligned} \alpha^i &= X_i(\alpha) + Y_i(\alpha)\sqrt{c} \ , \\ \overline{\alpha}^i &= X_i(\alpha) - Y_i(\alpha)\sqrt{c} \ . \end{aligned}$$

Очевидно, что $X_i(\alpha)$ и $Y_i(\alpha)$ являются целыми числами.

Предположим теперь, что выполнено

$$a^2 - cb^2 = 1$$

и рассмотрим определенные выше α и $\overline{\alpha}$. Тогда $\alpha\overline{\alpha} = 1$ и

$$X_i^2 - cY_i^2 = 1 \ ,$$

опуская аргумент α. Более того, для $j \geq i$

$$\begin{aligned} X_{i+j} &= 2X_iX_j - X_{j-i} \ , \\ Y_{i+j} &= 2X_iY_j - Y_{j-i} \ . \end{aligned}$$

Из этих соотношений, а также из

$$\begin{aligned} X_{i+j} &= X_iX_j + cY_iY_j \ , \\ Y_{i+j} &= Y_iX_j + X_iY_j \end{aligned}$$

мы получаем рекурсивные формулы

$$\begin{aligned} X_{2i} &= X_i^2 + cY_i^2 = 2X_i^2 - 1 \ , \\ Y_{2i} &= 2X_iY_i \ , \\ X_{2i+1} &= 2X_iX_{i+1} - X_1 \ , \\ Y_{2i+1} &= 2X_iY_{i+1} - Y_1 \ . \end{aligned}$$

Эти формулы предназначены для быстрого вычисления X_i и Y_i. Так как $X_0 = 1$ и $X_1 = a$, то $X_i(\alpha)$ также не зависит от b.

Естественным образом определим сравнения: запись

$$a_1 + b_1\sqrt{c} \equiv a_2 + b_2\sqrt{c} \pmod{n}$$

будет означать, что одновременно выполнено $a_1 \equiv a_2 \pmod{n}$ и $b_1 \equiv b_2 \pmod{n}$.

Если вместо равенства $a^2 - cb^2 = 1$ мы полагаем, что верно сравнение

$$(*) \qquad a^2 - cb^2 \equiv 1 \pmod{n} ,$$

то вышеуказанные рекурсивные формулы могут быть заменены на соответствующие сравнения по модулю n.

Основой для криптосистемы является следующая лемма. Эта лемма является аналогом теоремы Эйлера для RSA и обеспечивает взаимную обратимость процедур зашифрования и расшифрования. Доказательство леммы состоит из непосредственных вычислений и поэтому здесь опущено. Читатель, которого это заинтересует, может найти данное доказательство в [Wil].

Лемма 5.1 *Пусть n является произведением двух нечетных простых чисел p и q, a, b и c — целые числа, удовлетворяющие сравнению (*) и, кроме того, символы Лежандра $\delta_p = \left(\frac{c}{p}\right)$ и $\delta_q = \left(\frac{c}{q}\right)$ удовлетворяют сравнениям*

$$\delta_i \equiv -i \pmod{4} \text{ для } i = p \text{ и } i = q .$$

Пусть далее $(cb, n) = 1$ и символ Якоби $\left(\frac{2(a+1)}{n}\right)$ равен 1. Обозначим

$$m = (p - \delta_p)(q - \delta_q)/4$$

и полагаем, что e и d удовлетворяют сравнению

$$ed \equiv (m+1)/2 \pmod{m} .$$

При этих предположениях

$$\alpha^{2ed} \equiv \pm\alpha \pmod{n} ,$$

где $\alpha = a + b\sqrt{c}$.

Теперь мы в состоянии представить детали криптосистемы с открытым ключом, разработанной Уильямсом. Обсуждение будет состоять из трех составных частей: проектирования системы, зашифрования и расшифрования. В качестве иллюстрации будет использован пример Яркко Кари.

Проектирование системы. Сначала выбираются два огромных простых числа p и q и вычисляется их произведение n. Затем выбирается число c так, что символы Лежандра δ_p и δ_q удовлетворяют сравнениям

из леммы 5.1. Отметим, что c может быть найдено очень быстро с помощью подбора, так как примерно одно число из четырех удовлетворяет этим сравнениям. Следующее число s определяется (путем подбора) таким образом, чтобы символ Якоби

$$\left(\frac{s^2-c}{n}\right) = -1$$

(и $(s,n)=1$). Число m определяется, как и в лемме 5.1, а далее выбираются d, $(d,m)=1$, и e, удовлетворяющие сравнению из леммы.

Числа $n,\ e,\ c,\ s$ публикуются, в то время как $p,\ q,\ m,\ d$ держатся в секрете.

В качестве примера возьмем $p=11$ и $q=13$, произведением которых является $n=143$. Так как

$$\left(\frac{5}{11}\right) = 1 \equiv -11 \quad \text{и} \quad \left(\frac{5}{13}\right) = -1 \equiv -13 \pmod 4\ ,$$

мы можем выбрать $c=5$. Мы также можем выбрать $s=2$, потому что

$$\left(\frac{s^2-c}{n}\right) = \left(\frac{-1}{11}\right)\cdot\left(\frac{-1}{13}\right) = -1\cdot 1 = -1\ .$$

Далее получаем $m = 10\cdot 14/4 = 35$. В силу $23\cdot 16 \equiv 18 \pmod{35}$ мы можем наконец выбрать экспоненты шифровки и дешифровки $e=23$ и $d=16$.

Зашифрование. Исходными текстами являются числа w с $1<w<n$. (Если необходимо, исходный текст сначала делится на блоки подходящего размера.) Сперва w *кодируется* с помощью числа α, имеющего вид, описанный выше, а затем шифруется возведением α в степень e по модулю n. Рассмотрим сначала кодирование.

В зависимости от того, равен символ Якоби $\left(\frac{w^2-c}{n}\right)$ $+1$ или -1, полагаем

$$b_1 = 0 \qquad \text{и} \qquad \gamma = w + \sqrt{c}$$

или

$$b_1 = 1 \qquad \text{и} \qquad \gamma = (w+\sqrt{c})(s+\sqrt{c})$$

соответственно. (Случай, когда символ Якоби равен 0, так маловероятен, что мы его не рассматриваем.) В обоих случаях

$$\left(\frac{\gamma\overline{\gamma}}{n}\right) = 1\ .$$

В первом случае это очевидно, а во втором случае выполнено в силу выбора s.

Арифметика ведется по модулю n. Как и ранее, x^{-1} означает целое число, удовлетворяющее сравнению $xx^{-1} \equiv 1 \pmod{n}$. Эта запись будет использоваться и в том случае, когда x имеет вид $a + b\sqrt{c}$, при этом x^{-1} будет иметь тот же вид. Для удобства мы иногда будем писать xy^{-1} вместо x/y. Запись $(x, \mathrm{mod} n)$ распространяется и на числа вида $x = a + b\sqrt{c}$. Таким образом,

$$(a + b\sqrt{c}, \mathrm{mod} n) = (a, \mathrm{mod} n) + (b, \mathrm{mod} n)\sqrt{c} \ .$$

Кодирование завершается определением $\alpha = \gamma/\overline{\gamma}$. Итак, записав α в форме $\alpha = a + b\sqrt{c}$, получаем в первом случае (где $b_1 = 0$)

$$\alpha \equiv (w^2 + c)/(w^2 - c) + (2w/(w^2 - c))\sqrt{c} \pmod{n} \ .$$

Во втором случае (где $b_1 = 0$) мы имеем $\alpha \equiv a + b\sqrt{c} \pmod{n}$, где

$$a = \big((w^2 + c)(s^2 + c) + 4csw\big)/(w^2 - c)(s^2 - c)$$

и

$$b = \big(2s(w^2 + c) + 2w(s^2 + c)\big)/(w^2 - c)(s^2 - c) \ .$$

Определение $\alpha = \gamma/\overline{\gamma}$ гарантирует, что в обоих случаях будет выполняться

$$\alpha\overline{\alpha} = a^2 - cb^2 \equiv 1 \pmod{n}$$

и, следовательно, будет справедливо (*). Это, конечно же, можно проверить с помощью непосредственных вычислений. Мы заключаем также, что в обоих случаях

$$2(a + 1) \equiv 2((\alpha + \overline{\alpha})/2 + 1) \equiv \gamma/\overline{\gamma} + \overline{\gamma}/\gamma + 2 \equiv (\gamma + \overline{\gamma})^2/\gamma\overline{\gamma} \pmod{n} \ ,$$

откуда следует, что символ Якоби $\left(\frac{2(a+1)}{n}\right)$ равен 1, как это и требуется в лемме 5.1.

Имея закодированный исходный текст w в виде $\alpha = a + b\sqrt{c}$, мы можем теперь его зашифровать, возводя α в степень e по модулю n. Как это было показано ранее, результат данной операции может быть выражен с помощью X_e и Y_e. А последние могут быть вычислены быстро с помощью рекурсивных формул. Обозначим

$$E = (X_e(\alpha)Y_e^{-1}(\alpha), \mathrm{mod} n) \ .$$

Криптотекстом является тройка (E, b_1, b_2), где b_1 было определено выше, а b_2 равно 0 или 1 в зависимости от того, четно ли a или нечетно.

Число α^{2e} будет давать дополнительную информацию для расшифрования. Тем не менее, как будет показано ниже, оно может быть немедленно вычислено по данной информации. Биты b_1 и b_2 необходимы для того, чтобы результат расшифрования был однозначным. Без них, в целом, может быть получено четыре возможных варианта открытого текста.

Вернемся к нашему примеру. Возьмем исходный текст $w = 21$. Так как

$$\left(\frac{21^2-5}{143}\right) = \left(\frac{7}{11}\right)\cdot\left(\frac{7}{13}\right) = (-1)\cdot(-1) = 1\ ,$$

мы имеем $b_1 = 0$, $\gamma = 21+\sqrt{5}$ и

$$\alpha = \frac{21+\sqrt{5}}{21-\sqrt{5}} = \frac{446+42\sqrt{5}}{436} \equiv \frac{17+42\sqrt{5}}{7} \equiv 41(17+42\sqrt{5}) \equiv$$

$$\equiv 125+6\sqrt{5} \pmod{143}\ .$$

Алгоритмы для вычисления символа Якоби и $(w^2-c)^{-1}$ могут быть также объединены в один алгоритм.

Так как 125 нечетно, мы получаем $b_2 = 1$. Для вычисления $X_{23}(\alpha)$ и $Y_{23}(\alpha)$ мы используем рекурсивные формулы, к примеру, следующим образом.

$$\begin{array}{ll}
X_1(\alpha)\equiv 125\ , & Y_1(\alpha)\equiv 6\ ,\\
X_2(\alpha)\equiv 75\ , & Y_2(\alpha)\equiv 70\ ,\\
X_3(\alpha)\equiv 35\ , & Y_3(\alpha)\equiv 48\ ,\\
X_5(\alpha)\equiv 120\ , & Y_5(\alpha)\equiv 44\ ,\\
X_6(\alpha)\equiv 18\ , & Y_6(\alpha)\equiv 71\ ,\\
X_{11}(\alpha)\equiv 48\ , & Y_{11}(\alpha)\equiv 17\ ,\\
X_{12}(\alpha)\equiv 75\ , & Y_{12}(\alpha)\equiv 125\ ,\\
X_{23}(\alpha)\equiv 68\ , & Y_{23}(\alpha)\equiv 125\ .
\end{array}$$

Следовательно, мы получаем

$$E \equiv 68\cdot 125^{-1} \equiv 68\cdot 135 \equiv 28 \pmod{143}\ .$$

Таким образом, криптотекстом является тройка $(28, 0, 1)$.

Расшифрование. Используя первую компоненту E криптотекста, получатель может вычислить число α^{2e}:

$$\begin{aligned}\alpha^{2e} &\equiv \alpha^{2e}/(\alpha\overline{\alpha})^e \equiv \alpha^e/\overline{\alpha}^e = (X_e(\alpha)+Y_e(\alpha)\sqrt{c})/(X_e(\alpha)-Y_e(\alpha)\sqrt{c})\\ &\equiv (E+\sqrt{c})(E-\sqrt{c}) = (E^2+c)/(E^2-c)+(2E/(E^2-c))\sqrt{c} \pmod{n}.\end{aligned}$$

Отметим, что эти вычисления могут быть проделаны также и криптоаналитиком, который перехватил криптотекст. Тем не менее нужна секретная информация для вычисления

$$\alpha^{2e} = X_{2ed}(\alpha) + Y_{2ed}(\alpha)\sqrt{c} = X_d(\alpha^{2e}) + Y_d(\alpha^{2e})\sqrt{c}\ ,$$

где значения X_d и Y_d могут быть вычислены с помощью рекурсивных формул, так как известно α^{2e}. Теперь все предположения леммы 5.1 выполнены и, следовательно,

$$\alpha^{2ed} \equiv \pm\alpha \pmod{n}\ .$$

Последняя компонента b_2 криптотекста указывает, какой из знаков для α является верным. Итак, α найдено и исходный текст w теперь получается из α и b_1 (второй компоненты криптотекста) следующим образом. Полагаем

$$\alpha' = \begin{cases} \alpha, \text{ если } b_1 = 0\ , \\ \alpha(s-\sqrt{c})/(s+\sqrt{c}), \text{ если } b_1 = 1\ . \end{cases}$$

Тогда

$$\alpha' \equiv (w+\sqrt{c})/(w-\sqrt{c}) \pmod{n}\ ,$$

откуда получаем, что

$$w \equiv ((\alpha'+1)/(\alpha'-1))\sqrt{c} \pmod{n}\ .$$

Возвращаясь снова к численному примеру, мы вначале используем E для вычисления α^{2e}:

$$\begin{aligned} \alpha^{2e} &\equiv (28^2+5)/(28^2-5) + (2\cdot 28/(28^2-5))\sqrt{5} \\ &\equiv 95 + 126\sqrt{5} \pmod{143}\ . \end{aligned}$$

Напомним, что $d = 16$. Следовательно, мы вычисляем

$$\begin{aligned} X_1(\alpha^{2e}) &\equiv 95\ , & Y_1(\alpha^{2e}) &\equiv 126\ , \\ X_2(\alpha^{2e}) &\equiv 31\ , & Y_2(\alpha^{2e}) &\equiv 59\ , \\ X_4(\alpha^{2e}) &\equiv 62\ , & Y_4(\alpha^{2e}) &\equiv 83\ , \\ X_8(\alpha^{2e}) &\equiv 108\ , & Y_8(\alpha^{2e}) &\equiv 139\ , \\ X_{16}(\alpha^{2e}) &\equiv 18\ , & Y_{16}(\alpha^{2e}) &\equiv 137 \end{aligned}$$

и получаем, что $18 + 137\sqrt{5} \equiv \pm\alpha \pmod{143}$. Так как $b_2 = 1$, то a должно быть нечетным и

$$\alpha \equiv -(18 + 137\sqrt{5}) \equiv 125 + 6\sqrt{5} \pmod{143}\ .$$

Используя вторую компоненту $b_1 = 0$ криптотекста, заключаем, что $\alpha = \alpha'$ и поэтому

$$\begin{aligned} w &\equiv (126 + 6\sqrt{5})(124 + 6\sqrt{5})^{-1}\sqrt{5} \equiv \\ &\equiv (126 + 6\sqrt{5})(124 - 6\sqrt{5})9124^2 - 5 \cdot 6^2)^{-1}\sqrt{5} \equiv \\ &\equiv 83 \cdot 38^{-1} \equiv 21 \pmod{143}. \end{aligned}$$

Таким образом, получен $w = 21$, который и являлся первоначальным исходным текстом.

Читатель, проработавший все эти детали, наверняка согласится, что криптосистема Уильямса намного труднее в объяснении, чем RSA! Тем не менее отсюда не следует, что время зашифрования и расшифрования будет выше, чем в RSA. Здесь мы имеем также дополнительное преимущество в том, что предварительный криптоанализ с высокой вероятностью эквивалентен факторизации.

Криптоанализ в сравнении с факторизацией. Если найдено p или q, можно немедленно вычислить m и d. Обратно, пусть каким-нибудь способом криптоаналитик нашел алгоритм расшифрования. Этот алгоритм может быть использован для факторизации n следующим образом. Сначала путем подбора выбирается число x с

$$(**) \qquad \left(\frac{x^2 - c}{n}\right) = -1 \ .$$

Затем шифруется x, но процесс зашифрования мы начинаем с выбора $b_1 = 0$ и $\gamma = x + \sqrt{c}$. Таким образом, для символа Якоби используется ложное значение $+1$. Пусть $(E, 0, b_2)$ будет результирующим криптотекстом. Криптоаналитик теперь применяет свой алгоритм для нахождения соответствующего исходного текста w. Однако w не тот же самый, что и x, так как процесс зашифрования начался с обмана. Действительно, непосредственные вычисления покажут, что $(x - w, n)$ равно p или q. (Более подробно об этом см. [Wil].) Это означает, что криптоаналитик в состоянии факторизовать n. Ситуация является аналогичной той, когда известны два различных квадратных корня по модулю n.

Проделаем это для нашего примера. Выберем $x = 138$. Тогда будет выполнено условие (**). Ошибочно выбираем $b_1 = 0$ и $\gamma = 138 + \sqrt{5}$. Имеем

$$\alpha = \gamma/\overline{\gamma} = 73 + 71\sqrt{5} \ .$$

Так как 73 нечетно, мы получаем, что $b_2 = 1$. Вычисляем далее:

$$\begin{aligned} X_1(\alpha) &\equiv 73\ , & Y_1(\alpha) &\equiv 71\ , \\ X_2(\alpha) &\equiv 75\ , & Y_2(\alpha) &\equiv 70\ , \\ X_3(\alpha) &\equiv 9\ , & Y_3(\alpha) &\equiv 139\ , \\ X_5(\alpha) &\equiv 133\ , & Y_5(\alpha) &\equiv 44\ , \\ X_6(\alpha) &\equiv 18\ , & Y_6(\alpha) &\equiv 71\ , \\ X_{11}(\alpha) &\equiv 139\ , & Y_{11}(\alpha) &\equiv 82\ , \\ X_{12}(\alpha) &\equiv 75\ , & Y_{12}(\alpha) &\equiv 125\ , \\ X_{23}(\alpha) &\equiv 42\ , & Y_{23}(\alpha) &\equiv 73\ . \end{aligned}$$

Мы заключаем, что

$$E \equiv 42 \cdot 73^{-1} \equiv 28 \pmod{143}\ ,$$

откуда получаем криптотекст $(28, 0, 1)$. Расшифрование этого криптотекста было проведено выше и результатом являлся исходный текст $w = 21$. Теперь мы можем факторизовать n, так как

$$(x - w, n) = (117, 143) = 13\ .$$

Приведенное обсуждение показывает, что если криптоанализ проводится при условии “известен избранный криптотекст” (даже для *одного* выбранного криптотекста), система тут же вскрывается.

5.2. Итерация морфизмов

Предложено множество криптосистем, основанных на теории автоматов и формальных языков. Некоторые из них будут обсуждены в этом и следующем параграфах. Как мы уже отмечали ранее, целью этого обзора является представление разных подходов к построению криптосистем с открытым ключом, а не представление результирующих систем. Кроме вопросов безопасности такая оценка берет в расчет также и другие аспекты: легкость легальных операций, длину криптотекстов и т.д. Некоторые из этих аспектов будут отмечены ниже. Теоретико-числовые понятия будут объясняться только в том случае, когда это необходимо для понимания построения систем. Теория формальных языков будет использоваться без подробных объяснений, к примеру, при криптоанализе. Читатель, который заинтересуется теорией формальных языков, может обратиться к [Sa1].

Рассмотрим алфавиты Σ и Δ. Напомним, что Σ^* означает множество всех слов над Σ, включая пустое слово λ. Алфавиты Σ и Δ могут

быть одинаковыми, пересекаться или частично совпадать. Отображение $h : \Sigma^* \to \Delta^*$ называется *морфизмом*, если и только если равенство $h(xy) = h(x)h(y)$ выполнено для всех слов x и y над Σ. Из определения следует, что $h(\lambda) = \lambda$ и морфизм полностью определен значениями на буквах алфавита Σ. *Конечная подстановка* σ есть отображение Σ^* в множество конечных подмножеств Δ^*, такое, что $\sigma(xy) = \sigma(x)\sigma(y)$ выполнено для всех x и y над Σ. Сейчас будут сделаны два вывода относительно морфизмов. Пусть, к примеру, $\Sigma = \Delta = \{a, b\}$ и

$$\sigma(a) = \{a, ab\}, \qquad \sigma(b) = \{b, bb\} \ .$$

Тогда

$$\sigma(ab) = \{ab, abb, abbb\} \ .$$

Заметим, что $\sigma(ab)$ содержит только три элемента, поскольку слово abb получается двумя разными способами. Впоследствии иногда более удобным будет использование для морфизмов обозначения $(x)h$ вместо $h(x)$ и аналогично для конечных подстановок. Если L является языком, то

$$\sigma(L) = \{y | y \in \sigma(x), \text{ где } x \in L\} \ .$$

Приступим теперь к описанию криптосистемы. Рассмотрим два морфизма $h_0, h_1 : \Sigma^* \to \Sigma^*$, а также непустое слово w над Σ. Будем говорить, что четверка $G = (\Sigma, h_0, h_1, w)$ является *обратно детерминированной* тогда и только тогда, когда из условия

$$(w)h_{i_1} \dots h_{i_n} = (w)h_{j_1} \dots h_{j_m}$$

всегда следует

$$i_1 \dots i_n = j_1 \dots j_m \ .$$

Здесь все индексы i_t и j_t принадлежат множеству $\{0, 1\}$. Таким образом, обратная детерминированность означает, что для любой композиции морфизмов выход однозначно определяет порядок их применения; невозможны случаи, когда две различные последовательности морфизмов приводят к одинаковому результату.

Пример 5.1. Рассмотрим морфизмы, определенные следующим образом:

$$h_0(a) = ab, \quad h_0(b) = b, \quad h_1(a) = a, \quad h_1(b) = ba \ .$$

Если мы выберем $w = a$, то результирующая четверка не будет обратно детерминированной, потому что выход a получается с помощью последовательности единиц произвольной длины. Такое же утверждение верно и для $w = b$. С другой стороны, четверка $(\{a, b\}, h_0, h_1, ab)$ является обратно детерминированной. Это справедливо, так как последняя

буква слова открывает последний применяемый морфизм. Используя этот принцип, можно "раскрутить" слово w' обратно к исходному слову при условии, что w' было получено из w с помощью некоторой последовательности морфизмов.

□

Обратно детерминированные четверки G могут быть использованы как классические криптосистемы с помощью следующего очевидного способа. Последовательность битов $i_1 \dots i_n$ шифруется словом $(w)h_{i_1} \dots h_{i_n}$. Обратная детерминированность гарантирует взаимнооднозначность расшифрования. Так, если G является четверкой из примера 5.1 с $w = ab$, то зашифрование исходных текстов осуществляется следующим образом:

Исходный текст	Криптотекст
0	*abb*
1	*aba*
00	*abbb*
01	*ababa*
10	*abbab*
11	*abaa*
011	*abaabaa*

Конечно же, G должна храниться в секрете, если использовать ее в виде вышеописанной классической криптосистемы. Однако здесь нет различий между легальным расшифрованием и криптоанализом. Криптосистемы такого типа называются функциональными. В общем случае *функциональная криптосистема* задается двумя функциями f_0 и f_1 и начальным значением x. Последовательность битов $i_1 \dots i_n$ шифруется как $(x)f_{i_1} \dots f_{i_n}$. Для обеспечения взаимной однозначности расшифрования выполняется условие, соответствующее определенной выше обратной детерминированности. Если исходные тексты содержат более двух различных символов, то необходимо более двух функций.

Очевидным путем преобразования функциональной системы в криптосистему с открытым ключом является нахождение секретной лазейки, приводящей от открытых функций и значений к некоторым ситуациям с легким грамматическим разбором. Более подробно, мы знаем начальное значение x и функции f_0, f_1, как и значение y, где

$$y = (x)f_{i_1} \dots f_{i_n}$$

для некоторой композиции функций f_0 и f_1. По этой информации тяжело найти последовательность индексов $i_1, \dots i_n$, определяющих композицию, хотя мы и знаем, что такая последовательность единственна.

Тем не менее с помощью информации секретной лазейки уравнение может быть преобразовано к виду

$$y' = (x')g_{i_1} \dots g_{i_n} \ ,$$

где x', y', g_0, g_1 известны. Кроме того, теперь последовательность битов (тех же, что и в исходной последовательности) может быть легко найдена.

Рассмотрим, как строится секретная лазейка, когда обе функции являются морфизмами. В действительности секретная лазейка приводит к двум морфизмам с легким грамматическим разбором. Открываемая часть использует алфавит намного больший, чем Σ, и две конечные подстановки вместо двух морфизмов. Подстановки и начальное слово определяются таким образом, чтобы последовательность битов оставалась неизменной при переходе от "открытого" уравнения к уравнению, имеющему легкий грамматический разбор.

Более подробно, пусть $G = (\Sigma, h_0, h_1, w)$ является обратно детерминированной. Пусть алфавит Δ имеет мощность намного выше, чем Σ. Обычно Σ состоит из пяти букв, в то время как Δ — из 200 букв. Пусть морфизм $g : \Delta^* \to \Sigma^*$ отображает каждую букву в букву или пустое слово таким образом, что $g^{-1}(a)$ непусто для всех букв a из Σ. Это означает, что каждая буква d из Δ является либо *потомком* некоторой буквы из Σ, либо *пустышкой*. Буква d является потомком a, если $g(d) = a$. Из дополнительного условия, что $g^{-1}(a)$ не пусто, следует, что каждая буква из Σ имеет по крайней мере одного потомка. Буква d является пустышкой, если $g(d) = \lambda$.

Рассмотрим четверку $H = (\Delta, \sigma_0, \sigma_1, u)$, где σ_0 и σ_1 — конечные подстановки, определенные ниже, и u — слово над Δ, удовлетворяющее условию $g(u) = w$, т.е. u содержится в $g^{-1}(w)$. В общем случае u не единственно, потому что пустышки могут появляться где угодно и каждый из потомков может выбираться произвольно.

Конечные подстановки σ_0 и σ_1 также не единственны. Для каждого d в Δ $\sigma_0(d)$ является непустым конечным множеством слов y, таких, что если h_0 отображает $g(d)$ в x из Σ^*, то $g(y) = x$, т.е. $\sigma_0(d)$ есть конечное непустое подмножество множества $g^{-1}(h_0(g(d)))$. (Обычно мы пишем аргументы функций справа, как здесь. Нет никакой путаницы, что мы пишем их при шифровке слева — это нужно для сохранения надлежащего порядка в последовательности битов.) Подстановка σ_1 определяется аналогично, только с использованием h_1.

Четверка $H = (\Delta, \sigma_0, \sigma_1, u)$ открывается в качестве ключа зашифрования. Последовательность битов $i_1 \dots i_n$ шифруется выбором про-

извольного слова x из конечного множества

$$(u)\sigma_{i_1} \dots \sigma_{i_n} \ .$$

Если последовательность длинная, то она произвольным образом может быть поделена на блоки, которые затем шифруются по отдельности.

Все остальное, а именно Σ, h_0, h_1, w, g образует секретную лазейку. Существенным из них является "интерпретация" морфизма g: все остальные части могут быть вычислены с помощью g и открытой информации. Отметим заодно, что в терминологии L-систем G является *DTOL-системой*, а H — *TOL-системой*. *L-системы*, названные в честь А. Линденмайера, являются математическими моделями, которые очень подходят для компьютерного моделирования процессов биологического роста. Читатель может ознакомиться с этим подробнее в [RS].

Идея, лежащая в основе только что описанной криптосистемы, состоит в том, что криптоаналитик осуществляет разбор согласно беспорядочной TOL-системы H, в то время как легальный получатель, знающий секретную лазейку, может действовать легко и просто с помощью DTOL-системы G. Некоторые комментарии будут представлены ниже. Тот факт, что криптосистема с открытым ключом функционирует так, как и планировалось, вытекает из следующей леммы.

Лемма 5.2 *Пусть $G = (\Sigma, h_0, h_1, w)$ обратно детерминирована, а g и $H = (\Delta, \sigma_0, \sigma_1, u)$ определены, как и выше. Пусть G и H шифруют последовательности битов вышеописанным методом. Тогда расшифрование согласно H единственно. Кроме того, если последовательность битов $i_1 \dots i_n$ зашифровывается как y согласно H, то $i_1 \dots i_n$ расшифровывается как $g(y)$ согласно G.*

Доказательство. Рассмотрим последнее предложение. Полагаем, что y является словом из множества $(u)\sigma_{i_1} \dots \sigma_{i_n}$. Тогда

$$g(y) = (g(u))h_{i_1} \dots h_{i_n} \ .$$

Это следует из определения подстановок и u. (Знакомый с алгеброй читатель заметит, что подстановки, так же как и морфизмы, коммутируют согласно их строгому определению.)

Для доказательства единственности расшифрования согласно H предположим, что некоторое y может быть расшифровано одновременно и как последовательность битов i, и как последовательность битов j. По последнему предложению леммы $g(y)$ расшифровывается одновременно как i и j согласно G. Так как расшифрование согласно G однозначно в силу обратной детерминированности, мы имеем, что $i = j$.

□

Продолжая пример 5.1, полагаем $\Delta = \{c_1, c_2, c_3, c_4, c_5\}$ и определим интерпретацию морфизма g:

$$g(c_1) = b, \quad g(c_2) = g(c_4) = a, \quad g(c_3) = g(c_5) = \lambda \ .$$

Таким образом, c_2 и c_4 являются потомками a, c_1 — единственным потомком b, а c_3 и c_5 — пустышками. Выберем $u = c_4c_3c_1$, тогда $g(u) = ab = w$. Перед построением подстановок напомним, что морфизмы определялись следующим образом:

$$h_0 : \ a \to ab, \quad b \to b; \qquad h_1 : \ a \to a, \quad b \to ba \ .$$

Теперь определим σ_0 и σ_1, используя ту же самую наглядную запись.

$$\begin{array}{lll} \sigma_0 & c_1 \to c_1, c_3c_1 & \sigma_1 : \quad c_1 \to c_1c_2, c_3c_1c_4 \\ & c_2 \to c_4c_1, c_2c_1c_5 & \qquad\ \ c_2 \to c_2, c_3c_5c_4 \\ & c_3 \to c_5, c_3c_3 & \qquad\ \ c_3 \to c_3, c_5c_5 \\ & c_4 \to c_4c_1, c_2c_5c_1, c_4c_1c_3 & \qquad\ \ c_4 \to c_2, c_4c_3 \\ & c_5 \to c_5, c_3c_5c_3 & \qquad\ \ c_5 \to c_3, c_5c_3 \end{array}$$

Это определение верно, потому что после применения интерпретации морфизма g σ_0 и σ_1 преобразуются в h_0 и h_1:

$$\begin{array}{lll} h_0 & b \to b, b & h_1 : \quad b \to ba, ba \\ & a \to ab, ab & \qquad\ \ a \to a, a \\ & \lambda \to \lambda, \lambda & \qquad\ \ \lambda \to \lambda, \lambda \\ & a \to ab, ab, ab & \qquad\ \ a \to a, a \\ & \lambda \to \lambda, \lambda & \qquad\ \ \lambda \to \lambda, \lambda \end{array}$$

Шифруя 011 с помощью открытого ключа, мы сначала выбираем слово $y_1 = c_4c_1c_5c_1$ из $(u)\sigma_0$, затем слово $y_2 = c_2c_3c_1c_4c_3c_1c_2$ из $(y_1)\sigma_1$ и, наконец, слово

$$y_3 = c_2c_5c_5c_1c_2c_2c_3c_1c_2c_2$$

из $(y_2)\sigma_1$. Легальный получатель может вычислить

$$g(y) = abaabaa \ ,$$

откуда, используя отмеченное выше специальное свойство h_0 и h_1, немедленно получаем исходный текст 011.

□

Не все DTOL-системы, или четверки, $G = (\Sigma, h_0, h_1, w)$ являются обратно детерминированными. К примеру, если все слова $h_0(a)$, где a пробегает буквы алфавита Σ, являются степенями одного и того же слова x, то G не может быть обратно детерминированной. Это справедливо, так как легко проверить, что

$$(w)h_0h_1h_0h_0 = (w)h_0h_0h_1h_0 \ .$$

С другой стороны, обратная детерминированность не гарантирует простоту грамматического разбора. Для этой цели будет более подходящим понятие сильной обратной детерминированности.

По определению четверка $G = (\Sigma, h_0, h_1, w)$ является *сильно обратно детерминированной* тогда и только тогда, когда из условия

$$(w)h_{i_1} \dots h_{i_n} = (x)h_t$$

всегда следуют условия

$$t = i_n \quad \text{и} \quad x = (w)h_{i_1} \dots h_{i_{n-1}} \ .$$

Таким образом, каждое слово, полученное с помощью сильно обратно детерминированной G, имеет единственного предка в Σ^* и порождается из него с помощью единственного морфизма. Это означает, что грамматический разбор слова в сильно обратно детерминированной DTOL-системе зависит только от самого слова и, следовательно, разбор (расшифрование) может быть проведен справа налево без всякого предварительного просмотра. Это не обязательно верно, если G является только обратно детерминированной. Для того чтобы найти последний бит, может даже понадобиться вернуться к исходному пункту.

Пример 5.2. Очевидно, что каждая сильно обратно детерминированная DTOL-система является обратно детерминированной. Рассмотрим $G = (\{a, b\}, h_0, h_1, ab)$, где

$$h_0 : \ a \to ab, \quad b \to bb; \qquad h_1 : \ a \to bb, \quad b \to ab \ .$$

По индукции легко показать, что G обратно детерминирована: контрпример немедленно приводит к более короткому контрпримеру, который, конечно же, невозможен. С другой стороны, G не является сильно обратно детерминированной, потому что

$$(ab)h_0h_1 = bbababab = (abbb)h_1 = (baaa)h_0 \ .$$

Можно доказать, что сильная обратная детерминированность обладает свойством разрешимости, а обратная детерминированность — свойством неразрешимости.

□

Важным пунктом при создании криптосистемы является длина слова. Криптотексты не должны быть намного длиннее по сравнению с исходными текстами. Удачно, что существуют большие классы DTOL-систем с линейным ростом. При переходе к TOL-системам рост будет существенно тем же для потомков букв. Замены для пустышек должны определяться таким образом, чтобы не появлялся экспоненциальный рост. Кроме того, для уменьшения роста всегда можно использовать деление исходного текста на блоки.

Что касается предварительного криптоанализа, то он, вероятно, не будет успешным. Рассмотрим *секретную пару* (G, g), такую, что G является DTOL-системой, полученной из H с помощью g. Для заданного H существует несколько таких секретных пар. Только одна из них, скажем (G_1, g_1), будет использоваться разработчиком криптосистемы. Если найдется другая пара (G_2, g_2), порожденная H, то она может быть использована для расшифрования со следующим предупреждением. G_2 не обязательно является обратно детерминированной и, следовательно, один и тот же криптотекст может приводить к нескольким исходным текстам. Тем не менее правильный исходный текст всегда находится среди них.

Это наблюдение не делает предварительный криптоанализ существенно более легким. Можно показать, что нахождение *любой* секретной пары с помощью предварительного криптоанализа является NP-полной задачей. (Могут еще существовать некоторые другие методы.) Поэтому и нахождение пустых символов является NP-полной задачей. Если пустышки будут найдены, то построение секретной пары будет легким. Этот результат означает, что нет большой пользы от знания того, что пустышки всегда заменяются словами из пустых символов.

Подводя итог, можно сказать, что криптосистема является защищенной против предварительного криптоанализа: секретные пары не могут быть легко найдены. Криптоаналитический алгоритм имеет время работы kn^3 , где n — длина перехваченного криптотекста и k — достаточно большая константа, которая может быть найдена с помощью использования теории конечных автоматов [Kar2].

В следующем *обобщении* системы для успешного криптоанализа не достаточно нахождения пустых символов.

Напомним, что выше для интерпретации морфизма было предложено считать его значениями только буквы или пустое слово. Такое очень ограничивающее определение не является необходимым. Теперь мы положим, что интерпретацией морфизма является любой *сюръективный* морфизм $g : \Delta^* \to \Sigma^*$. Это означает, что все слова из Σ^* появля-

ются как значения g, что непременно удовлетворяет нашему первоначальному определению. В противном случае, создание криптосистемы остается неизменным. Тем не менее будем более подробными.

Как и ранее, полагаем, что $G = (\Sigma, h_0, h_1, w)$ обратно детерминирована (предпочтительнее даже сильно обратно детерминирована). Выберем Δ намного больше Σ и пусть морфизм $g : \Delta^* \to \Sigma^*$ сюръективен. Пусть u есть слово над Δ, такое, что $g(u) = w$. Так как g сюръективно, такое u всегда существует. Для d из Δ пусть $\sigma_i(d)$, $i = 0, 1$, будет конечным непустым множеством слов x, для которых

$$g(x) = h_i(g(d)) \ .$$

Вновь, для того чтобы гарантировать существование таких слов x, необходима сюръективность g. Как и ранее, расшифрование криптотекста y осуществляется с помощью грамматического разбора слова $g(y)$ согласно G.

Лемма 5.2 остается верной и сейчас. Предварительный криптоанализ, кажется, будет очень трудным. Тем не менее вышеупомянутый алгоритм для анализа перехваченных криптотекстов с кубическим временем работы применим и для обобщенной системы.

Пример 5.3. Рассмотрим $G = (\{a, b\}, h_0, h_1, ba)$, где морфизмы определены следующим образом:

$$\begin{array}{lll} h_0 & a \to ab & \qquad h_1 : \quad a \to ba \\ & b \to b & \qquad\qquad\ \ b \to a \end{array}$$

Тогда G сильно обратно детерминирована по очевидным причинам: последняя буква слова определяет используемый морфизм и предок слова единствен в силу инъективности обоих морфизмов. Выберем $\Delta = \{c_1, \ldots, c_{10}\}$ из десяти букв и определим интерпретацию морфизма g:

$$\begin{array}{lll} g & c_1 \to ab & \qquad c_6 \to \lambda \\ & c_2 \to b & \qquad c_7 \to bab \\ & c_3 \to \lambda & \qquad c_8 \to \lambda \\ & c_4 \to b & \qquad c_9 \to ba \\ & c_5 \to a & \qquad c_{10} \to aa \end{array}$$

Мы можем выбрать $u = c_9$, потому что $g(c_9) = ba$. Завершая определе-

ние H, зададим подстановки σ_0 и σ_1:

$$
\begin{array}{lll}
\sigma_0 & c_1 \to c_1c_4 & \sigma_1: \quad c_1 \to c_4c_{10}, c_9c_5 \\
& c_2 \to c_6c_2 & \qquad\quad c_2 \to c_8c_5 \\
& c_3 \to c_3c_6 & \qquad\quad c_3 \to c_6c_8 \\
& c_4 \to c_2, c_3c_4 & \qquad\quad c_4 \to c_3c_5 \\
& c_5 \to c_1, c_5c_2 & \qquad\quad c_5 \to c_9, c_2c_5 \\
& c_6 \to c_8c_3 & \qquad\quad c_6 \to c_6c_3 \\
& c_7 \to c_7c_8c_4 & \qquad\quad c_7 \to c_1c_{10} \\
& c_8 \to c_8 & \qquad\quad c_8 \to c_8 \\
& c_9 \to c_3c_7, c_4c_1 & \qquad\quad c_9 \to c_1c_6c_5, c_5c_9 \\
& c_{10} \to c_5c_7, c_1c_5c_2 & \qquad\quad c_{10} \to c_9c_9, c_7c_5
\end{array}
$$

Это определение верно, потому что для всех i и d $\sigma_i(d)$ содержит только слова x, удовлетворяющие $g(x) = h_i(g(d))$. К примеру, из первой и двух последних строк мы получаем

$$
\begin{array}{ccccccccc}
g(c_4c_{10}) & = & g(c_9c_5) & = & baa & = & h_1(ab) & = & h_1(g(c_1)) \ , \\
g(c_3c_7) & = & g(c_4c_1) & = & bab & = & h_0(ba) & = & h_0(g(c_9)) \ , \\
g(c_5c_7) & = & g(c_1c_5c_2) & = & abab & = & h_0(aa) & = & h_0(g(c_{10})) \ .
\end{array}
$$

Исходный текст 01101 шифруется согласно окрытому ключу H, например, следующим образом:

$$
\begin{array}{lll}
c_9 & \to & c_4c_1 \to c_3c_5c_9c_5 \to c_6c_8c_9c_5c_9c_9 \\
& \to & c_8c_3c_8c_3c_7c_1c_4c_1c_4c_1 \\
& \to & c_8c_6c_8c_8c_6c_8c_1c_{10}c_4c_{10}c_3c_5c_9c_5c_3c_5c_4c_{10} = y \ .
\end{array}
$$

При легальном расшифровании сначала вычисляется

$$g(y) = abaabaaabaaabaa \ .$$

С помощью грамматического разбора G далее получают уравнения

$$
\begin{array}{l}
h_1(bababbabbab) = g(y) \ , \\
h_0(baababa) = bababbabbab \ , \\
h_1(abaa) = baababa \ , \\
h_1(bab) = abaa \ , \\
h_0(ba) = bab \ ,
\end{array}
$$

где индексы h дают исходный текст 01101.

В обобщенной версии криптосистемы, где интерпретация морфизма выбирается более свободно, пустые символы несущественны для безопасности, как это имело место в основной версии. Наоборот, небрежное использование “пустышек” может нанести ущерб безопасности.

В вышеприведенной иллюстрации некоторые криптоаналитические заключения могут быть основаны на первых шести символах криптотекста y. Этот вывод будет очевиднее, если при разработке криптосистемы мы выберем $u = c_9c_3$ вместо $u = c_9$. Тогда из любого криптотекста немедленно отделяется суффикс z, порожденный буквой c_3 в u. Это так, потому что c_3 порождает только буквы c_3, c_6, c_8 и, кроме того, последняя буква слова, порожденного с помощью c_9, отлична от трех упомянутых. В z все появления c_8 могут быть проигнорированы и грамматический разбор может основываться на морфизмах s_0 и s_1, определяемых с помощью σ_0 и σ_1.

$$\begin{array}{lll} s_0 & c_3 \to c_3c_6 & \qquad s_1: \quad c_3 \to c_6 \\ & c_6 \to c_3 & \qquad\quad\ \ c_6 \to c_6c_3 \end{array}$$

К примеру, $z = c_3c_3c_3c_3c_6c_3c_3c_3c_3c_6c_3$ может быть проанализировано следующим образом:

$$\begin{array}{l} s_0(c_6c_6c_6c_3c_6c_6c_6c_3c_6) = z\ , \\ s_1(c_3c_3c_6c_3c_3c_6c_3) = c_6c_6c_6c_3c_6c_6c_6c_3c_6\ , \\ s_0(c_6c_3c_6c_3c_6) = c_3c_3c_6c_3c_3c_6c_3\ , \\ s_1(c_6c_6c_3) = c_6c_3c_6c_3c_6\ , \\ s_1(c_3c_6) = c_6c_6c_3\ , \\ s_0(c_3) = c_3c_6\ . \end{array}$$

Индексы s дают исходный текст 011010. Во многих случаях этот алгоритм расшифрования может быть даже более легким, чем легальное расшифрование с помощью G!

□

5.3. Автоматы и теория формальных языков

В обсуждаемой в предыдущем параграфе криптосистеме лежащая в ее основе задача принадлежит теории формальных языков. Криптоанализ и легальное расшифрование равносильны грамматическому разбору криптотекста. Этот разбор будет существенно более легким, если известна секретная лазейка. Мы обсудили данную криптосистему достаточно подробно. Криптосистемы, представленные в этом параграфе, также базируются на теории формальных языков и близких ей областях. Однако наше обсуждение будет теперь более кратким и поверхностным. Все детали из основных областей не будут объясняться. В самом деле, такое объяснение не является целью данного обзора.

Достаточно много из предложенных криптосистем с открытым ключом используют идею, которая может быть сформулирована как *зашифрование с помощью раскрашивания*. Краска связывается с каждой буквой исходного текста. Так как мы вновь полагаем, что исходные тексты являются двоичными последовательностями, то мы используем только две краски — *белую* (*бит 0*) и *черную* (*бит 1*). Открытый ключ зашифрования дает метод, который порождает произвольно много элементов, раскрашенных белой краской, и произвольно много элементов, раскрашенных черной краской. В обсуждаемых здесь криптосистемах такими элементами являются слова. Биты шифруются как слова, раскрашенные соответствующим образом. Конечно, для различных появлений бита 0 (соответственно 1) должны выбираться различные слова, раскрашенные белой (соответственно черной) краской. В идеальной ситуации задача определения краски для заданного слова является труднорешаемой, в то время как знание секретной лазейки позволяет очень легко решать эту задачу. Очевидно, что открытый ключ зашифрования всегда дает метод решения, который может иметь экспоненциальную сложность: по очереди порождаются слова обоих цветов до тех пор, пока они не встретятся в криптотексте.

Все криптосистемы, основанные на зашифровании с помощью раскрашивания, имеют тенденцию к недопустимо большому росту длины слова. Число слов, пригодных для зашифрования каждого бита, должно быть огромным, предпочтительнее даже потенциально бесконечным. В противном случае криптоаналитик может породить все зашифрования битов заранее и использовать таблицу для расшифрования. Это приводит к несколько парадоксальной ситуации, когда повышение безопасности зависит от роста отношения между длиной криптотекста и длиной исходного текста.

Типичная криптосистема с открытым ключом, основанная на идее шифрования с помощью раскрашивания, использует задачу тождества слова для *конечно порожденных групп*, т.е. групп с конечным множеством образующих $\{a_1, \ldots, a_m\}$ и конечным числом отношений вида

$$(*) \qquad w_i = \lambda, \quad i = 1, \ldots, n \ ,$$

где λ является единицей группы и каждое w_i является словом над алфавитом $\{a_1, a_1^{-1}, \ldots, a_m, a_m^{-1}\}$. Два слова x и y над этим алфавитом называются *эквивалентными* тогда и только тогда, когда существует конечная последовательность слов

$$x = x_0, x_1, \ldots, x_t = y,$$

такая, что для $j = 0, \ldots, t-1$, x_{j+1} получается из x_j добавлением или

удалением появления w_i, w_i^{-1}, zz^{-1} или $z^{-1}z$, где $1 \leq i \leq n$ и z — произвольное слово. *Задача тождества слова* для конечно порожденной группы состоит в проверке для произвольного слова эквивалентности с единицей λ. (Очевидно, что x и y эквивалентны тогда и только тогда, когда xy^{-1} эквивалентно λ.) Существуют специальные группы с неразрешимой задачей тождества слова.

Такая группа G используется в качестве базиса для построения криптосистемы с открытым ключом следующим образом. Полагаем, что G задана с помощью (*). Соотношения (*), так же как и два неэквивалентных слова y_0 и y_1, публикуются в качестве ключа зашифрования. Шифровка бита i есть произвольное слово, эквивалентное y_i. Таким образом, зашифровывая бит 0, отправитель применяет в произвольном порядке к y_0 правила преобразования, заданные с помощью (*). Это, конечно же, может быть сделано заранее: отправитель порождает огромное количество шифровок обоих битов и держит их в безопасном месте до тех пор, пока они не понадобятся.

Расшифрование, по крайней мере в принципе, сводится к задаче тождества слова: надо решить, эквивалентно ли слово y_0 или y_1. Метод зашифрования гарантирует, что каждое слово будет эквивалентно точно одному из y_0 и y_1.

Секретная *лазейка* состоит из дополнительно определенных отношений, таких, что задача тождества слов будет легкой для построенной группы G', но слова y_0 и y_1 еще не являются эквивалентными. Таким образом, G' имеет одинаковые образующие с G, но в дополнение к (*) выполнены некоторые другие отношения. Можно всегда выбрать y_0 и y_1 в качестве образующих G, чтобы новые отношения не давали тождества этих образующих. Новые отношения могут, например, вводить некоторую коммутативность для того, чтобы задача тождества слов в G' становилась легкой.

Применимость криптосистемы зависит от выбора данных отношений. В частности, составляющие секретную лазейку дополнительные отношения существенны с точки зрения безопасности системы. Важно отметить, что для криптоаналитика не обязательно находить эти дополнительные отношения, действительно используемые при создании системы. Для этого подходят любые отношения, такие, что задача тождества слов легка, и для результирующей группы G'' не эквивалентны два открытых слова. Это означает, что разработчик криптосистемы должен быть очень осторожен при подборе секретной лазейки.

Например, если G имеет две образующие a и b, которые открываются, то введение дополнительного отношения $bab^{-1}a^{-2} = \lambda$ порождает коммутативность в виде $ba = a^2b$. В зависимости от других отношений это может сделать задачу тождества слов существенно более простой.

Подобные криптосистемы могут быть построены с помощью использования отношений, связанных с алгебраическими структурами, отличными от групп. Не известны результаты относительно сложности предварительного криптоанализа (для условия — "известен только ключ зашифрования").

Следующая криптосистема с открытым ключом основана на *регулярных языках*. Как и для систем, основанных на повторяющихся морфизмах, предварительный криптоанализ является NP-полным. (Заметим отличие от основной рюкзачной системы.) Система использует также пустые буквы, как и при зашифровании с помощью раскрашивания. Здесь необходимы некоторые сведения из теории автоматов. Они не будут объясняться, а читатель может найти их в [Sa1]. Язык, порожденный с помощью грамматики G (соответственно принимаемый автоматом A), обозначим через $L(G)$ (соответственно $L(A)$).

Представим сначала упрощенную версию системы. Выберем две произвольные грамматики G_0 и G_1 с одним и тем же терминальным алфавитом Σ и конечный детерминированный автомат A_0 над Σ. Переведем все заключительные состояния автомата A_0 в незаключительные и наоборот. Обозначим полученный автомат через A_1. Итак, языки $L(A_0)$ и $L(A_1)$ являются дополнениями друг друга. Построим такую грамматику G_i', что

$$L(G_i') = L(G_i) \cap L(A_i), \text{ для } i = 0, 1 \ .$$

Грамматики G_i' легко получаются из G_i и A_i с помощью конструкции, стандартной в теории формальных языков.

Грамматики G_0' и G_1' теперь *открываются* в качестве ключа зашифрования. Используется зашифрование с помощью раскрашивания: бит i шифруется как произвольное слово в $L(G_i')$. Автоматы A_i хранятся в качестве секретной лазейки. Перехватчик должен определять принадлежность к $L(G_i')$, что в общем случае неразрешимо. Легальный получатель расшифровывает, решая легкую задачу определения принадлежности криптотекста к $L(A_0)$ или $L(A_1)$. Разработка системы гарантирует, что расшифрование всегда является однозначным.

Теперь мы готовы описать полную версию криптосистемы. В действительности сложностные результаты, отмеченные ниже, касаются полной системы.

Сначала определим *морфический образ грамматики*. Рассмотрим грамматику G и h — морфизм, отображающий каждую терминальную букву G в терминальную букву или пустое слово и каждую нетерминальную букву из G в нетерминальную букву. Морфический образ $h(G)$ грамматики G состоит из всех продукций $h(\alpha) \to h(\beta)$, где $\alpha \to \beta$ —

продукция в G. Начальная буква $h(G)$ совпадает с морфическим образом первой буквы из G. Для грамматик G_1 и G_2 запись $G_1 \subseteq G_2$ означает, что множество продукций G_1 является подмножеством множества продукций G_2.

В полной версии криптосистемы A_i и G_i' определяются, как и выше. Пусть Δ — алфавит, намного больший Σ, $G_i''(i = 0, 1)$ — грамматики с терминальным алфавитом Δ, а h является таким морфизмом, что

$$h(G_i'') \subseteq G_i' \ .$$

Пара (G_0'', G_1'') составляет открытый ключ зашифрования. Как и ранее, используется зашифрование с помощью раскрашивания. Расшифрование всегда будет однозначным. Для расшифрования легальный получатель применяет морфизм h к одному блоку криптотекста. Если x есть результирующее слово, то соответствующий бит исходного текста равен 0 тогда и только тогда, когда A_0 принимает x.

Можно показать, что предварительный криптоанализ является NP-полной задачей в следующем смысле. NP-полным будет поиск по заданной паре (G_0'', G_1'') таких (h, A_0), что первая пара является результатом второй пары, а также G_0, G_1 с помощью процесса, описанного выше. Другими известными криптосистемами с открытым ключом, имеющими NP-полный предварительный криптоанализ, являются система, рассмотренная в примере 2.2, и система, основанная на повторяющихся морфизмах. Конечно, это свойство не гарантирует безопасности: данный результат ничего не говорит о сложности криптоанализа в целом.

Система, основанная на регулярных языках, может быть усилена даже с помощью замены грамматик G_0'' и G_1'' на некоторые эквивалентные грамматики. Рассматривается специальный метод построения эквивалентных грамматик. Не ясно, дает ли это существенное усиление.

Наконец, будут кратко отмечены некоторые криптосистемы с открытым ключом, основанные на теории автоматов. *Последовательная машина* M является конечным автоматом с выходом. M переводит входное слово в выходное слово той же длины. Инверсия M^{-1} переводит выходное слово обратно во входное. Пусть k натуральное число. Инверсия $M^{-1}(k)$ *с задержкой* k действует следующим образом. Пусть для входного слова x выходом будет y. После получения на входе слова yu, где u — произвольное слово длины k, $M^{-1}(k)$ порождает на выходе слово wx, где w — слово длины k. В системе с открытым ключом k и $M^{-1}(k)$ открываются в качестве ключа зашифрования, а M хранится как секретная лазейка. Зашифрование исходного текста y происходит с помощью выбора произвольного слова u длины k и применения $M^{-1}(k)$ к слову yu. При расшифровании криптотекста c легальный получатель игнорирует первые k букв c и применяет M к оставшемуся слову. В

целом, трудно вычислить M по заданным k и $M^{-1}(k)$. Для данной криптосистемы неизвестны какие-либо оценки сложности.

Клеточные автоматы CA являются перспективной основой для криптосистем с открытым ключом. Случай еще как следует не изучен. К примеру, по заданному обратимому клеточному автомату очень трудно найти его инверсию. Не существует алгоритмов для нахождения инверсии клеточного автомата с временной сложностью, ограниченной вычислимой функцией. Открытый ключ зашифрования составляют обратимый клеточный автомат CA и натуральное число k. Исходный текст кодируется как конфигурация CA и шифруется применением CA k раз к данной конфигурации. Результирующая конфигурация криптотекста образует криптотекст. Инверсия CA^{-1} образует секретную лазейку. Для расшифрования легальный получатель применяет CA^{-1} k раз к криптотексту.

5.4. Теория кодирования

Рассмотрим ситуацию, когда при передаче информации могут появляться ошибки. Шум в канале связи может быть вызван технической неисправностью, а также небрежностью со стороны отправителя или получателя. В теории кодирования полагается, что ошибки появляются случайно и независимо друг от друга. Вероятность ошибочной передачи нуля единицей, а также единицы — нулем считается одинаковой. При этой постановке полагается, что нет противника, действующего преднамеренно. Целью теории кодирования является введение *избыточности* таким образом, чтобы даже при возникновении ошибок сообщение правильно бы восстанавливалось. Конечно же, что при этом будут сделаны некоторые ограничения на число исправляемых ошибок, так как для исправления неограниченного числа ошибок нельзя определить достаточную меру избыточности.

Более подробно, предположим, что при передаче n-разрядного слова w могут возникать d ошибок. Таким образом, принятое слово w' отличается от w не более чем в d разрядах, при этом неверными могут быть разряды в любой части слова. Пусть $C = \{\alpha_1, \ldots, \alpha_k\}$ — множество n-разрядных слов α_i, любые два из которых различаются не менее чем в $2d + 1$ разрядах. Тогда C назовем *кодом, исправляющим* d *ошибок*. Действительно, α_i может быть правильно восстановлено из полученного α_i', так как α_i и α_i' отличаются не более чем в d разрядах, тогда как α_i' отличается от любого α_j, где $j \neq i$, не менее чем в $d+1$ разрядах.

Хотя *декодирование*, т. е. восстановление α_i из α_i', всегда в принципе

возможно, но оно может быть трудновычислимым. Действительно, описанные ниже криптосистемы с открытым ключом используют идею, что знание только открытого ключа приводит к трудновычислимой задаче декодирования, в то время как знание секретной лазейки позволяет получателю легко декодировать сообщения.

Таким образом, теория кодирования и криптография преследуют противоположные цели. В теории кодирования пытаются представить сообщение в таком виде, чтобы при передаче сообщения было бы допустимо приемлемое количество ошибок. В этом смысле ясность сообщения повышается. С другой стороны в криптографии стараются уменьшить ясность, чтобы сообщение было непонятно для перехватчика. Так как эти цели противоположны, трудно объединить два данных подхода, хотя очень важно приводить сообщения к виду, защищающему одновременно и от перехватчика, и от случайного шума в канале связи. Для этих целей сообщение должно быть сначала зашифровано, после чего к результату применяется самокорректирующееся кодирование. Обратный порядок этих двух операций не имеет смысла по понятным причинам. Защита одновременно против шума и активного перехватчика кажется невозможной. В целом детали комбинирования криптографии с теорией кодирования еще как следует не продуманы.

Такое комбинирование не имелось в виду для кратко описанных ниже криптосистем с открытым ключом. Система, разработанная Макэлисом, имеет сходство с рюкзачными системами, в частности основанными на плотных рюкзаках. Криптосистема использует *коды Гоппы*, исправляющие d ошибок. Такой код $\{\alpha_1, \ldots, \alpha_k\}$ основан на полиномах степени d, неприводимых над конечным полем $F(2^m)$. Его можно задать с помощью порождающей $k \times n$ булевой матрицы M, где $n = 2^m$. Создатель системы выбирает произвольные булевы матрицы S и P, такие, что S — невырожденная $k \times k$-матрица и P — перестановочная $n \times n$-матрица. Затем находится булева матрица $M' = SMP$, где все числа берутся по модулю 2.

Важнейшим является тот факт, что M' по крайней мере выглядит как порождающая матрица для произвольного *линейного кода*. Для линейных кодов не известны декодирующие алгоритмы, которые не являются трудновычислимыми, в то время как коды Гоппы легко декодируются вручную.

Создатель криптосистемы публикует M' в качестве ключа зашифрования, а S, M и P хранит в качестве секретной лазейки. С помощью информации секретной лазейки также легко вычисляются S^{-1} и P^{-1}.

Блок исходного текста w из k разрядов кодируется как

$$c = wM' \oplus b \ ,$$

где b — произвольный двоичный n-разрядный вектор, имеющий ровно d единиц, а $\oplus$ означает побитовое сложение. Векторы b выбираются для каждого блока w по отдельности.

Легальный получатель знает, что $M' = SMP$ и вычисляет

$$cP^{-1} = wSM \oplus bP^{-1} \ .$$

Так как P — перестановочная матрица, то в точности d компонент в bP^{-1} равны 1. Следовательно, вектор ошибок bP^{-1} может быть устранен с помощью техники декодирования кодов Гоппы, что дает wS. Отсюда, умножая на S^{-1}, тут же получаем w.

Перехватчик рассматривает задачу декодирования линейного кода. Декодирование линейных кодов является NP-полной задачей.

Предварительный криптоанализ безнадежен, если n и d достаточно велики: существует очень много вариантов выбора матриц M, S и P. К примеру, если $n = 1024$ и $d = 50$, то существует приблизительно 10^{149} полиномов, которые могут использоваться для построения кодов Гоппы. Параметры связаны формулой $k = 2^m - md = n - md$. Полагая, что обращение $k \times k$-матрицы требует k^3 операций, временная сложность предварительного криптоанализа может быть грубо оценена как

$$\frac{k^3 \binom{n}{k}}{\binom{n-d}{k}} \ .$$

Для $n = 1024$ и $d = 50$ это дает величину $2^{80.7}$. Можно показать, что вероятность существования более одной секретной лазейки весьма мала.

Глава 6

Криптографические протоколы

6.1. Больше, чем просто этикет

Термин “протокол” обычно связывают с обычаями и предписаниями дипломатических формальностей, табелях о ранге и этикете. Например, протоколом определяются места за столом или порядок выступлений. И бывает так, что на международной встрече значительная часть времени тратится на согласование протокола размещения участников.

Криптографический протокол образует алгоритм обмена информацией между различными участниками, как соперничающими между собой, так и нет. Такой алгоритм использует криптографические преобразования и обычно базируется на криптографии с открытым ключом. Однако цель протокола состоит не только в обеспечении секретности при передаче сообщений. Так, например, участники могут для достижения какой-то общей цели раскрыть друг другу часть своих секретов или объединить свои усилия для раскрытия секрета неизвестного каждому из них в отдельности. Или два участника, удаленные друг от друга, пожелают сгенерировать совместно какую-нибудь случайную последовательность. Один участник может также хотеть убедить другого, что он обладает некоторой информацией, не раскрывая при этом никакую часть этой информации. Протоколы, реализующие эти цели, существенно изменили наши представления о том, что является невозможным, когда несколько участников, соперничающих или нет, обмениваются информацией между собой.

Протоколы создаются для реализации конкретной цели. При оценке

безопасности протокола следует принимать во внимание как безопасность используемой в протоколе криптосистемы, так и самого протокола. Интуитивно ясно, что безопасность протокола может быть, самое большее, такой же, как и для используемой криптосистемы, хотя может быть и гораздо меньше.

Рассмотрим, к примеру, следующую весьма общую постановку задачи. Требуется установить защищенный канал связи между двумя индивидуальными пользователями информационной системы или сети связи. Никаких предположений о том, общались между собой эти пользователи или нет, не делается. Основная идея, лежащая в основе криптографии с открытым ключом, может быть использована и для решения поставленной задачи. Полученный протокол очень простой и состоит из следующих двух шагов. Вначале все пользователи публикуют свои ключи для зашифрования. Затем сообщения, направляемые к пользователю A, шифруются при помощи его ключа зашифрования. В этом случае используемая криптосистема может быть безопасной, тем не менее сам протокол не предотвращает возможности обмана: пользователь C, посылая сообщения к A, может притвориться пользователем B. Для предотвращения появления таких ситуаций к протоколу следует добавить соглашение о подписи передаваемых сообщений.

Отделяя свойства используемой криптосистемы от протокола, необходимо иметь в виду возможный тип противника. В большинстве коммуникационных протоколов противник относится к одному из следующих трех типов.

1. Сами участники протокола, пытающиеся хитрить. Позже у нас встретятся два способа хитрить: пассивный и активный.

2. Пассивные перехватчики. Они являются достаточно безобидными, но могут стать обладателями информации, не предназначающейся для них.

3. Активные перехватчики. Помимо получения секретной информации, они могут испортить весь протокол.

В простом протоколе, приведенном выше, пользователь C, пытающийся притвориться B, относится к типу (3). В качестве примера противника типа (1) рассмотрим протокол из примера 2.3 для игры в покер по телефону. Предположим также, что зашифрование и расшифрование выполняются как модульное возведение в степень и взятие дискретного логарифма, соответственно. (В примере 2.3 конкретные способы зашифрования и расшифрования не были указаны.) Более точно, предположим, что игроки A и B пришли к соглашению о выборе

большого безопасного простого p и о представлении карт числами из интервала $(2, p-1)$. Каждый игрок выбирает экспоненты зашифрования и расшифрования, удовлетворяющие условию

$$e_A d_A \equiv e_B d_B \equiv 1 \pmod{p-1} ,$$

после чего зашифрование и расшифрование выполняются по модулю p. Отметим, что свойство быть квадратичным вычетом (по модулю p) сохраняется при зашифровании. И если раздающий карты заметит, что численное представление каждого туза является квадратичным вычетом, то только квадратичные невычеты будут розданы противнику и только квадратичные вычеты самому сдающему. Сдающий теперь знает, что у противника нет тузов. Ясно, что в таком случае раздача карт не является равновероятной. И даже в случае модуля n, когда задача поиска квадратичных вычетов является трудной, раздающий карты может проследить за движением полных квадратов (по mod n) во время выполнения протокола.

Обычно достаточно трудно получить математические результаты о безопасности протокола как такового или по отношению к безопасности используемой криптосистемы. Необходимо тщательно анализировать те методы, посредством которых была установлена безопасность протоколов, а также посредством которых была доказана невозможность того или иного протокола, удовлетворяющего тем или иным требованиям. С тем же самым мы сталкиваемся и при анализе обычных алгоритмов. Однако криптографические протоколы отличаются от обыкновенных алгоритмов в том смысле, что каждый участник протокола обладает какими-то вычислительными возможностями (которые могут варьироваться от одного участника к другому) и способен делать выводы. Это означает, что каждый участник может соотнести априорное и уже полученное знание с информацией, содержащейся в только что полученном сообщении. И новое отправляемое им сообщение зависит от всей этой информации.

Следующая модификация популярной игры иллюстрирует эти замечания. Участники игры (их число произвольное) образуют цепь. Первый и последний участник знают свои позиции. Кроме того, каждый член цепи знает следующего участника в цепи. В течение первой фазы протокола первый участник посылает число 2 второму участнику, и каждый последующий участник добавляет 1 к получаемому им числу и посылает результат дальше, за исключением последнего участника, который ничего дальше не посылает. После завершения этой фазы каждый участник знает свой порядковый номер i. Вторая фаза протокола

состоит в преобразовании сообщения w, состоящего из цепочки английских букв. Сообщение w первоначально находится у участника 1. Получив сообщение w', участник i применяет систему Цезаря с ключом i к первой букве w', переносит результат в конец слова и посылает образованное таким способом новое слово следующему участнику. Последний же участник опять ничего не посылает. Если в игре участвует семь человек, то, например, исходный текст $w = \mathrm{SAUNA}$ преобразуется так:

$$\mathrm{SAUNA} \to \mathrm{AUNAT} \to \mathrm{UNATC} \to \mathrm{NATCX} \to \mathrm{ATCXP} \to \mathrm{TCXRF} \to \mathrm{CXRFZ}$$

Получив слово CXRFZ, седьмой участник может немедленно расшифровать сообщение, так же как, впрочем, и все остальные члены. Ясно, что этот протокол уязвим для противника любого из трех типов (1)–(3).

Протокол для отправки шифрсообщений, в котором получатель подтверждает получение, может быть описан следующим образом. Ключи зашифрования E_A, $E_B, \ldots$ открываются, в то время как каждый из ключей расшифрования D_A, $D_B, \ldots$ известен только соответствующему пользователю. Согласно протоколу, отправка сообщения w от A к B выполняются за два шага.

Шаг 1: A посылает B тройку $(A, E_B(w), B)$.

Шаг 2: Используя D_B, B расшифровывает сообщение и подтверждает его получение, посылая A тройку $(B, E_A(w), A)$.

Активный перехватчик C может теперь перехватить тройку на шаге 1 и направить B тройку $(C, E_B(w), A)$. Не заметив этот трюк, B посылает C на шаге 2 тройку $(B, E_C(w), C)$, после чего C может расшифровывать! Даже следующая версия, в которой используются подписи, не меняет существенно ситуацию.

Шаг 1: A посылает B пару $(E_B(E_B(w)A), B)$.

Шаг 2: Используя D_B, B определяет A и w и подтверждает получение, посылая A пару $(E_A(E_A(w)B), A)$.

Здесь функции E и D предполагаются определенными на числах. Имена A, $B, \ldots$ — это последовательности цифр и $E_B(w)A$ есть последовательность цифр, полученная конкатенацией $E_B(w)$ с A.

Активный перехватчик C может в этом случае перехватить пару $(E_A(E_A(w)B), A)$ на шаге 2. Таким образом, C знает $E_A(w')$, где $w' = E_A(w)B$. C посылает теперь A пару $(E_A(E_A(w')C), A)$ и A, думая, что это есть шаг 1 протокола, подтверждает получение, посылая C пару $(E_C(E_C(w')A), C)$. Теперь C, применив D_C, узнает w', а также $E_A(w)$, после чего C опять посылает A, но уже пару $(E_A(E_A(w)C), A)$. После

подтверждения от A, т. е. получив пару $(E_C(E_C(w)A), C)$, C может вычислить w. Конечно, A может узнать о перехвате и быть более осторожным, сохраняя посланные и полученные сообщения.

В заключение этого параграфа приведем схему подписи, основанную на удостоверении личности i пользователя сети. Удостоверение i может быть связано, например, с именем пользователя и сетевым адресом. Оно должно удостоверять пользователя и быть доступным другим пользователям. Удостоверение выполняет функции открытого ключа зашифрования.

Предлагаемая схема предполагает также наличие заслуживающего доверия административного органа, единственной целью которого является передача каждому пользователю секретного числа x, построенного по удостоверению i пользователя. Это происходит, когда пользователь подключается к сети.

Более точно, пусть n, e и d выбраны также, как и в криптосистеме RSA, и f есть односторонняя функция двух переменных. Значения n, e и функция f открываются, а d и разложение n известно только административному органу. Секретное число, передаваемое административным органом пользователю с удостоверением i, есть число

$$x = (i^d, \mathrm{mod} n) \ .$$

Подпись этого пользователя сообщения w есть любая пара (s,t), такая, что

$$(*) \qquad s^e \equiv i \cdot t^{f(t,w)} \pmod{n} \ .$$

По данной тройке (w,s,t) это условие может быть проверено остальными пользователями, поскольку i, n, e и f являются открытой информацией.

С другой стороны, пользователь может порождать свою подпись (s,t) для сообщения w, выбирая случайное число r и вычисляя

$$t = (r^e, \mathrm{mod} n) \quad \text{и} \quad s = (xr^{f(t,w)}, \mathrm{mod} n) \ .$$

Поскольку $x^e \equiv i \pmod{n}$, проверочное условие (*) выполняется. Функция f используется для криптографического хеширования t и w.

6.2. Подбрасывание монеты по телефону. Игра в покер

В некоторых криптографических протоколах требуется, чтобы участники, расположенные, возможно, далеко друг от друга, породили совместно какую-нибудь случайную последовательность, не обращаясь

за помощью к третьей стороне. Если участников двое, то это равносильно *подбрасыванию монеты по телефону*. Ясно, что A и B не могут сделать это обычным образом, когда один из них бросает монету и сообщает по телефону результат другому. В этом случае потребовалась бы абсолютная честность участников. С другой стороны, если один из участников находится под наблюдением надежного рефери, это подбрасывание монеты не представляет никаких проблем.

Хорошо известная иллюстрация М.Блюма этой ситуации состоит в следующем. A (Алиса) и B (Боб) недавно развелись и живут в разных городах. Они хотят решить с помощью жребия (подбрасывая монету), кому из них достанется автомобиль. При этом третьей стороны — рефери, которому бы оба полностью доверяли, нет. Цель, которую хотят достичь A и B, может быть также проиллюстрирована с помощью следующей модели: *бросание монеты в колодец*.

A и B стоят достаточно далеко друг от друга. B стоит около глубокого колодца с очень чистой водой. Когда B бросает монету в колодец (не теряя ее из виду!), то он видит результат брошенного жребия, но уже не в состоянии изменить его. С другой стороны, A еще не знает результата. Поэтому B просто сообщает его A. Результат жребия, возможно, используется для каких-то целей. Позже A может подойти к колодцу и посмотреть в воду, чтобы проверить, что информация, полученная от B, была верной. В принципе здесь не важно, кто именно, A или B, бросает монету в колодец. В приведенной модели схвачена суть подбрасывания монеты по телефону. Оно едва ли осуществимо, если только один из участников играет активную роль.

Следующий протокол показывает, каким образом A бросает монету B по телефону. Вначале только B знает результат жребия и сообщает его A. (Этот результат может инициировать выполнение некоторой части какого-то другого протокола.) Позже A может проверить, что полученная от B информация о результате жребия была верной.

Шаг 1: A выбирает два больших простых числа p и q и сообщает B их произведение $n = pq$.

Шаг 2: B выбирает случайное число u из интервала $(1, n/2)$ и сообщает A его квадрат $z = (u^2, \mathrm{mod} n)$.

Шаг 3: A вычисляет $\pm x$ и $\pm y$ — четыре квадратичных корня из z $(\mathrm{mod}\ n)$. Это возможно, поскольку A знает разложение n на два простых множителя. Пусть x' будет меньшим из двух чисел $(x, \mathrm{mod} n)$ и $(-x, \mathrm{mod} n)$. Пусть y' определяется аналогично для $\pm y$. Заметим, что $u = x'$ или $u = y'$.

Шаг 4: A выбирает наугад $u = x'$ или $u = y'$. Более того, A находит наименьшее число i, такое, что i-й разряд справа в двоичном представлении x' отличается от соответствующего разряда в y' и сообщает B наугад один из вариантов "i-й разряд справа в твоем числе и есть 0" или "i-й разряд справа в твоем числе и есть 1".

Шаг 5: B сообщает A, была ли его догадка верной ("орел") или нет ("решка").

Шаг 6: Позже B разрешает A "подойти к колодцу", сообщая число u.

Шаг 7: A сообщает разложение числа n.

Ясно, что у A нет способа узнать u, так что ему действительно приходится гадать. Если бы B мог схитрить, сменив число u после сделанного A угадывания, то тогда это означало бы, что в его распоряжении есть оба числа x' и y' и,следовательно, он может разложить n, вычислив наибольший общий делитель $x' - y'$ и n. В частности, чтобы избежать этого, A на шаге 4 сообщает только один бит, а не сообщает x' или y' целиком.

Протокол опирается на предположение, что разложение на множители является трудновычислимой задачей. На самом деле, можно сказать, что значительная часть теории криптографических протоколов опасно зависит от сложности одной этой задачи.

Для бросания монеты по телефону было предложено много других протоколов. Ниже дается одна общая схема. A и B оба знают (предположительно) одностороннюю функцию f, но не ее обратную f^{-1}. B выбирает случайное число x и сообщает A значение $f(x)$. A делает предположение о некотором равновозможном свойстве числа x. (Например: x четное или нечетное.) B сообщает A, была ли догадка верной. Позже B сообщает A само число x.

Следующий протокол отличается в некоторой степени от вышеприведенного, хотя и основывается на схожих идеях и предположениях. В нем используются некоторые теоретико-числовые факты относительно символа Якоби $\left(\frac{a}{n}\right)$ (см. приложение Б). Предположим, что $n = pq$, как и ранее. Рассмотрим число a, такое, что $0 < a < n$ и $(a, n) = 1$. В точности половина из таких чисел a удовлетворяет равенству $\left(\frac{a}{n}\right) = 1$ и опять ровно половина из чисел, удовлетворяющих последнему равенству, является квадратичными вычетами по модулю n. Значение символа Якоби легко вычисляется. Является ли a квадратичным вычетом по модулю n может быть легко проверено, только если известны p и q.

Протокол протекает следующим образом. B выбирает p и q и сообщает A их произведение n, а также случайное число a, такое, что $\left(\frac{a}{n}\right) = 1$. A угадывает, является ли a квадратичным вычетом по модулю n или нет. B сообщает A, была или нет его догадка верной. Позже B позволяет A "подойти к колодцу", раскрывая p и q.

Здесь существенным является то, что A проверяет, что раскрытые числа p и q есть в действительности простые числа. (Это замечание принадлежит Юхе Хонкала.) В противном случае B мог бы схитрить следующим образом. Для начала B выбирает три простых числа p_1, p_2, q_1 и число a, такое, что

$$\left(\frac{a}{p_1}\right) = \left(\frac{a}{p_2}\right) = -1 \quad \text{и} \quad \left(\frac{a}{q_1}\right) = \pm 1 \ .$$

Допустим, что правильное угадывание A интерпретируется как "орел", а неправильное — как "решка".

Если B выбрал орла, он действует следующим образом. Если A говорит "вычет", то B открывает $p = p_1 p_2$ и $q = q_1$. Если же A говорит "невычет", то B открывает $p = p_1$, $q = p_2 q_1$.

Если B выбирает "решку", то он действует так. Если A говорит "вычет", B открывает $p = p_1$, $q = p_2 q_1$. Если A говорит "невычет", то B открывает $p = p_1 p_2$, $q = q_1$.

Подбрасывание монеты может быть естественным образом расширено на случай, когда A *подбрасывает число* x к B. Это означает подбрасывание x бит за битом. После этого процесса B знает x, а A знает только число разрядов в двоичной записи x.

Эта идея с подбрасыванием числа может быть применена для получения более безопасного протокола, [GM], для игры в покер по телефону. 52 карты представляются шестиразрядными двоичными числами. Любой полиномиальный алгоритм, угадывающий с вероятностью не менее 51 процента конкретный бит карты оппонента, может быть преобразован к вероятностному полиномиальному алгоритму для факторизации $n = pq$. Протокол использует теоретико-числовой факт, что $\left(\frac{a}{n}\right) = -\left(\frac{b}{n}\right)$, как только a и b являются различными квадратными корнями одного и того же числа по модулю n, и, более того, $p \equiv q \equiv 3 \pmod 4$. ("Различные" означает здесь, что $a \not\equiv \pm b \pmod n$.) Дадим теперь краткое описание протокола.

Для каждого $i = 1, \ldots, 52$ A выбирает два больших простых числа p_i и q_i, удовлетворяющих условию $p_i \equiv q_i \equiv 3 \pmod 4$, и вычисляет их произведение $n_i = p_i q_i$. Затем A тасует колоду карт и присваивает число n_i i-й карте в перетасованной колоде. Сама i-я карта представляется шестеркой $(t_1, t_2, \ldots, t_6)$, где каждое t_i есть случайное число

с $\left(\frac{t_j}{n_i}\right) = 1$ и, более того, t_j есть квадратичный вычет по модулю n_i в точности в том случае, когда j-й бит двоичного представления i-й карты равен 1. A сообщает B все 52 шестерки вместе с числами n_i (но не сообщает p_i, q_i).

B перемешивает свою колоду, порождает и сообщает A аналогичное представление своих карт, используя большие простые числа r_i, s_i и их произведения m_i, где $i = 1, \ldots, 52$.

Чтобы сдать одну карту B, A подбрасывает 52 числа x_i к B. Таким образом, B знает x_i для всех $i = 1, \ldots, 52$, а A не знает в этот момент ни одного из них. B сообщает A числа $(x_i^2, \mathrm{mod} n_i)$ вместе со значениями $\left(\frac{x_i}{n_i}\right)$, за исключением одного конкретного значения $i = k$, для которого B сообщает A число $(x_k^2, \mod n_k)$ и значение $-\left(\frac{x_k}{n_k}\right)$. A может вычислить квадратные корни, поскольку ей известно разложение n_i. A сообщает теперь B конкретное значение квадратного корня из x_i^2, чей символ Якоби был сообщен ей B. Заметим, что у A нет возможности узнать выделенное значение $i = k$. Но для данного значения B получил достаточно информации, чтобы разложить n_k. Действительно, B знает два различных квадратных корня из одного и того же числа по модулю n_k. Таким образом, B может расшифровать k-ю карту из колоды A. Это и будет одна из его карт. Для значений $i \neq k$ B не получил никакой дополнительной информации. Наконец, B вынимает из своей колоды сданную ему карту и сообщает A, какая карта была удалена из колоды. A не в состоянии расшифровать эту информацию и, таким образом, не знает, какая карта была удалена.

После этого B сдает A одну карту из своей колоды, следуя той же самой процедуре. Поскольку в его колоде осталась только 51 карта, он подкидывает 51 число к A. Эта процедура продолжается, пока не будет сдано по пять карт обоим участникам. (Процедура без труда модифицируется для других разновидностей покера.) После игры вся секретная информация раскрывается. Нетрудно видеть, как на этом этапе может быть обнаружено возможное шулерство. Например, можно обнаружить, что игрок взял на одном шаге более, чем одну карту, хотя временно можно заявить более одного символа Якоби. Учитывая все детали подбрасывания чисел, работу с квадратичными вычетами и т.д., можно сказать, что протокол в целом является очень громоздким.

6.3. Как разделить секрет

Этот параграф является отступлением от основной темы в том смысле, что здесь не будет протоколов. Тем не менее развиваемая здесь техника

используется также и во многих криптографических протоколах.

Будем говорить, что t участников A_i, где $i = 1, \ldots, t$, *k-разделяют секрет* c, $1 < k \leq t$, если и только если выполнены следующие условия (1)–(3).

1. Каждый участник A_i знает некоторую информацию a_i, неизвестную остальным участникам A_j, где $j \neq i$.

2. Секрет c легко может быть вычислен по любым k значениям a_i.

3. Знание любых $k-1$ значений a_i, неважно каких, не позволяет определить секрет c.

Множество $\{a_1, \ldots, a_t\}$, удовлетворяющее условиям (2)–(3), будем называть (k, t) *пороговой схемой* для секрета c. Возможные ограничения на c будут рассмотрены в примере 6.1.

Практическая значимость пороговых схем очевидна. Например, c может содержать в себе инструкции для выполнения некоторых особо важных заданий. Для начала выполнения этих заданий необходимо согласие по крайней мере k участников. С другой стороны, любые из k участников в состоянии начать выполнение этих действий совершенно независимо от того, согласны или нет другие участники.

Можно привести много примеров из различных областей математики, когда некоторый объект определяется k фактами из некоторого набора фактов, когда любые дополнительные факты будут избыточными. Такие примеры могут быть использованы при построении пороговых схем. Возможно, один из самых простых и в то же время легко описываемых примеров основывается на модульной арифметике и китайской теореме об остатках.

Пусть m_i, $i = 1, \ldots, t$, целые числа, большие 1, такие, что $(m_i, m_j) = 1$ при $i \neq j$. Пусть a_i, $i = 1, \ldots, t$, тоже целые, $0 \leq a_i < m_i$. (На самом деле с таким же успехом a_i могли бы быть произвольными целыми.) Пусть M обозначает произведение всех m_i. Обозначим $M_i = M/m_i$ и пусть N_i будет обратным к $M_i \pmod{m_i}$, $i = 1, \ldots, t$. Таким образом, $M_i N_i \equiv 1 \pmod{m_i}$. Так как $(M_i, m_i) = 1$, то обратный элемент существует и легко находится из алгоритма Евклида.

Сравнения

$$x \equiv a_i \pmod{m_i}, \qquad i = 1, \ldots, t,$$

обладают общим решением

$$x = \sum_{i=1}^{t} a_i M_i N_i \ .$$

Более того, это решение единственно в том смысле, что любое другое решение y удовлетворяет равенству

$$(y, \mathrm{mod} M) = (x, \mathrm{mod} M) \ .$$

(Заметим, что это дает также и доказательство китайской теоремы об остатках. Ясно, что любые два решения должны быть сравнимы по модулю M. Очевидно также, что x является решением, поскольку M_i делится на все m_j, где $j \neq i$.)

Пусть теперь k фиксировано, $1 < k \leq t$. Обозначим через $\min(k)$ наименьшее из произведений k различных множителей m_i. Таким образом, $\min(k) = m_1 \ldots m_k$, если m_i следуют в возрастающем порядке. Аналогично обозначим через $\max(k-1)$ наибольшее из произведений $k-1$ множителей m_i. Предположим, что выполнено

$$(*) \qquad \min(k) - \max(k-1) \geq 3 \cdot \max(k-1) \ .$$

(Желательно, чтобы m_i выбирались так, чтобы искомая разность была бы большой.) Пусть c целое, удовлетворяющее условиям

$$\max(k-1) < c < \min(k) \ .$$

Определим числа a_i

$$a_i = (c, \mathrm{mod} m_i), \qquad i = 1, \ldots, t.$$

Теорема 6.1 *Множество $\{a_1, \ldots, a_t\}$ образует (k, t) пороговую схему для c.*

Доказательство. Предположим вначале, что известны любые k из a_i-х. Не теряя общности, считаем, что это $a_1, \ldots, a_k$. Обозначим $M' = m_1 \ldots m_k$, $M'_i = M'/m_i$, $i = 1, \ldots, k$, и пусть N'_i будет обратным к $M'_i \pmod{m_i}$. Определяя

$$y = \sum_{i=1}^{k} a_i M'_i N'_i \ ,$$

по китайской теореме об остатках получаем

$$y \equiv c \pmod{M'} \ .$$

Поскольку $M' \geq \min(k) > c$, получаем

$$c = (y, \mathrm{mod} M') \ ,$$

откуда видно, как можно вычислить c с помощью чисел $a_1, \ldots, a_k$.

Предположим теперь, что известны только $k-1$ значений a_i-х. Не теряя общности, считаем, что это $a_1, \ldots, a_{k-1}$. Определим y как и раньше, только используя на этот раз модули $m_1, \ldots, m_{k-1}$, тогда получаем

$$y \equiv c \pmod{m_1 \ldots m_{k-1}} .$$

Однако здесь для c в силу (*) имеется много возможностей. Действительно, всего имеется

$$(**) \qquad \left[\frac{(\min(k) - \max(k-1) - 1)}{m_1 \ldots m_{k-1}}\right]$$

возможностей, что является громадным числом в случае, если m_i выбраны большими и близкими друг к другу.

□

Пример 6.1. Выберем $k = 3$, $t = 5$ и

$$m_1 = 97, \quad m_2 = 98, \quad m_3 = 99, \quad m_4 = 101, \quad m_5 = 103 .$$

Тогда $\min(k) = 941094$, $\max(k-1) = 10403$ и (**) изменяется между 89 и 97, в зависимости от выбора двух значений m_i. Наибольшее значение 97 получается для произведения $m_1 m_2 = 9506$, а наименьшее значение 89 — для произведения 10403.

Секрет c — это число, удовлетворяющее неравенствам

$$10403 < c < 941094 .$$

Предположим, что некое агентство знает c и раздало участникам A_i значения

$$a_1 = 62, \quad a_2 = 4, \quad a_3 = 50, \quad a_4 = 50, \quad a_5 = 38 .$$

Относительно модуля m_i можно предполагать, что он общедоступен, либо предполагать, что он известен только участнику A_i. В последнем варианте центральное агентство, распределяющее секрет c, передавая частичную информацию участникам A_i, должно также позаботиться о том, чтобы m_i удовлетворяли требуемым условиям.

Предположим теперь, что участники A_2, A_3, A_4 желают объединить свои знания, чтобы выяснить c. Вначале они вычисляют

$$M_1' = 9999, \ M_2' = 9898, \ M_3' = 9702, \ N_1' = 33, \ N_2' = 49, \ N_3' = 17 .$$

Таким образом,

$$y = 4 \cdot 9999 \cdot 33 + 50 \cdot 9898 \cdot 49 + 50 \cdot 9702 \cdot 17 = 33816668$$

и соответственно

$$c = (y, \mathrm{mod} 98 \cdot 99 \cdot 101) = (y, \mathrm{mod} 979902) = 500000 \ .$$

Аналогично если A_1, A_4, A_5 захотят узнать c, то они вычисляют

$$M_1' = 10403, \ M_2' = 9991, \ M_3' = 9797, \ N_1' = 93, \ N_2' = 63, \ N_3' = 43 \ ,$$

$$y = 62 \cdot 10403 \cdot 93 + 50 \cdot 9991 \cdot 63 + 38 \cdot 9797 \cdot 43 = 107463646 \ ,$$

$$c = (y, \mathrm{mod} 97 \cdot 101 \cdot 103) = 500000 \ .$$

С другой стороны, A_2 и A_5 вдвоем узнают, что

$$y = 4 \cdot 103 \cdot 59 + 38 \cdot 98 \cdot 41 = 176992 \equiv 5394 \equiv c \pmod{10094} \ ,$$

после чего A_2 и A_5 знают только, что c есть одно из чисел вида

$$5394 + i \cdot 10094, \qquad 1 \leq i \leq 92 \ ,$$

даже если они знают все модули m_i. Правильным значением i является $i = 49$. Аналогично если A_3 и A_4 объединят свои знания, то они найдут

$$y = 50 \cdot 101 \cdot 50 + 50 \cdot 99 \cdot 50 = 500000 \equiv 50 \pmod{9999} \ .$$

Это говорит им лишь о том, что c есть одно из чисел вида

$$50 + i \cdot 9999, \qquad 2 \leq i \leq 94 \ .$$

Конечно, у них не было никакого способа узнать, что они на самом деле попали в цель и нашли верное значение c, когда вычисляли y!

□

6.4. Частичное раскрытие секретов

Быстро развивающаяся область с широким спектром приложений образует вопросы следующего типа. Два или более участников обладают какими-то секретами. Для достижения общей цели они намерены разделить друг с другом часть своей информации, но не слишком большую. Для этой цели и следует создать протокол.

При разделении секретов, обсуждаемом в параграфе 6.3, участники, для того чтобы получить информацию c, должны были раскрыть свои секреты полностью. Сейчас же все участники сотрудничают, но раскрывают свои секреты только частично. (Более того, мы не предполагаем наличие центрального агентства. Правда, такое агентство не нужно и в параграфе 6.3 в случае, если модули предаются гласности.)

Обшая постановка задачи частичного раскрытия секретов может быть сформулирована следующим образом. Каждый из участников $A_1, \ldots, A_t, t \geq 2$, знает функцию $f(x_1, \ldots, x_n)$. Здесь каждая переменная меняется внутри некоторого начального отрезка множества натуральных чисел и сама функция принимает натуральные значения. Таким образом, функция f может быть задана таблицей. Каждый из участников A_i знает конкретные значения a_i из области определения переменной x_i , но не обладает никакой информацией в отношении значений a_j для $j \neq i$. Участники $A_1, \cdots, A_n$ желают вычислить значение функции $f(a_1, \ldots, a_n)$, не раскрывая никакой дополнительной информации о своих собственных значениях a_i. Другими словами, протокол должен быть построен так, чтобы после его выполнения все участники A_i знали бы значения $f(a_1, \ldots, a_n)$, но никто из них не раскрыл бы никакой дополнительной информации о значении a_i. Тут под дополнительной понимается любая информация, которая не извлекается непосредственно из значения функции $f(a_1, \ldots, a_n)$. Конечно, все это становится тривиальным, если разрешается использовать непредвзятого арбитра.

Для примера рассмотрим

$$f(x_1, x_2, x_3) = \begin{cases} 1, & \text{если некоторое } x_i \text{ не является простым} \\ & \text{наименьшее простое среди аргументов иначе.} \end{cases}$$

Если $a_2 = 19$ и $f(a_1, a_2, a_3) = 17$, то A_2 знает, что одно из чисел a_1 и a_3 равно 17, а другое является простым ≥ 17. Если $a_2 = 4$ и $f(a_1, a_2, a_3) = 1$, то A_2 не получает никакой информации о числах a_1 и a_3.

Для решения поставленной проблемы созданы различные протоколы. Оказалось, что вопросы безопасности здесь трудно формализуемы в общем случае, в частности при возможности коллективного обмана, когда некоторые участники образуют коалиции с целью перехитрить остальных. С другой стороны, эти протоколы открывают совершенно новые перспективы для конфиденциального обмена информацией. Например, могут быть реализованы новые схемы для тайного голосования. Скажем, некоторые члены совета могут обладать правом вето. С новыми протоколами никто не будет знать, получено ли отрицательное решение на основе решения большинства или кто-то использовал свое право вето или же по обеим этим причинам.

Рассмотрим конкретный пример. Участники $S_1, S_2, P_1, \ldots, P_t$, где t — нечетное, желают принять или отвергнуть какое-то решение. Все участники могут голосовать "за" или "против". В дополнение к этому, S_1 и S_2 имеют возможность использовать "суперголоса" S-за и S-против. Заранее оговорено, что в случае отсутствия суперголосов вопрос решается мнением большинства. В случае одного или двух одинаковых суперголосов все остальные обыкновенные голоса игнорируются. В случае двух противоположных суперголосов вопрос решается большинством обыкновенных голосов. Такое голосование можно представить как происходящее в ООН с S_1 и S_2 в качестве сверхдержав. Например, предположим, что голоса распределились в соответствии со следующей таблицей:

S_1	S_2	P_1	P_2	P_3	P_4	P_5
S-за	против	против	за	за	за	за

После выполнения протокола все участники знают результат. При этом S_1 не знает, что полученный результат был бы тем же самым, если бы он проголосовал против. А S_2 не знает, что не смог бы изменить результат голосования. Участники P_i не знают, что их голоса не влияли на принятие решения. После же выполнения протокола с голосами

S_1	S_2	P_1	P_2	P_3	P_4	P_5
S-за	S-против	за	за	против	против	против

S_1 знает, что S_2 проголосовал S-против и что большинство "рядовых стран" P_i проголосовали против.

Указанная проблема тайного голосования с суперголосами может быть непосредственно сформулирована в терминах общей задачи вычисления значений функций. Это же относится и к следующей проблеме, для которой мы детально опишем протокол.

A(Алиса) и B(Боб) желают выяснить, кто из них старше, не узнав при этом ничего больше о возрасте другого. Каким образом они могут совершить диалог, удовлетворяющий этому требованию?

Более точно, мы хотим построить протокол для следующей беседы. Вначале A знает i и B знает j, где i и j — целые, выражающие их возраст в годах. В конце беседы они оба знают, выполнено ли $i \geq j$, или же $i < j$, но A и B не получили никакой дополнительной информации о j и i соответственно.

Рассматриваемую проблему часто формулируют в таком виде: два миллионера хотят узнать, кто из них богаче, не давая никакой дополнительной информации о своем состоянии.

Будем предполагать, что рассматриваемые возрасты могут лежать в пределах от 1 до 100, т. е. i и j могут быть целыми от 1 до 100.

Приводимый протокол основан на криптосистеме с открытым ключом. Таким образом, B знает ключ зашифрования Алисы E_A, но не знает ее ключа расшифрования D_A.

Шаг 1: B выбирает большое случайное число x и вычисляет $E_A(x) = k$.

Шаг 2: B сообщает A число $k - j$.

Шаг 3: A вычисляет числа

$$y_u = D_A(k - j + u) \text{ для } 1 \leq u \leq 100 \ .$$

После этого A выбирает большое случайное простое p. (Приблизительный размер p несколько меньше размера x. Приблизительные размеры p и x согласуются заранее.) Далее A вычисляет числа

$$z_u = (y_u, \mathrm{mod} p), \quad 1 \leq u \leq 100 \ .$$

Она проверяет, что для всех u и всех $v \neq u$

$$(*) \qquad |z_u - z_v| \geq 2 \text{ и } 0 < z_u < p - 1 \ .$$

Если это не так, A выбирает другое простое число до тех пор, пока не будет выполняться условие (*).

Шаг 4: A сообщает B последовательность чисел (в указанном порядке)

$$(**) \qquad z_1, \ldots, z_i, z_{i+1} + 1, z_{i+2} + 1, \ldots, z_{100} + 1, p \ .$$

Шаг 5: B проверяет, сравнимо или нет j-е число из последовательности с $x \pmod p$. Если да, он заключает, что $i \geq j$. Если нет, он заключает, что $i < j$.

Шаг 6: B сообщает A результат.

Заключение, сделанное на шаге 5, будет верным, поскольку j-е число z'_j в (**) удовлетворяет условиям

$$i \geq j \text{ влечет } z'_j = z_j \equiv y_j = x \pmod p \text{ и}$$

$$i < j \text{ влечет } z'_j = z_j + 1 \not\equiv z_j \equiv y_j = x \pmod p \ .$$

Условие (*) гарантирует, что в последовательности (**) нет повторов. (Добавление 1 не играет здесь роли, поскольку положительная разность любых двух z не меньше 2.) Если k-е число уже появлялось раньше в (**), то B знает, что $i < k$.

Необходимо перемешивание чисел y_u по модулю p. Если бы A просто сообщила B последовательность

$$y_1, \ldots, y_i, y_{i+1} + 1, y_{i+2} + 1, \ldots, y_{100} + 1 \ ,$$

то тогда B смог бы найти i, применяя E_A к этой последовательности, поскольку B известны все числа $k - j + u$, $1 \leq u \leq 100$.

Единственная информация, которая поступает от B к A (соответственно от A к B), дается на шаге 2 (соответственно на шаге 4), и эта информация ничего не дает. С другой стороны, здесь всегда возможен обман, который очень трудно обнаружить без беспристрастного арбитра или действительного раскрытия значений i и j. Например, невозможно проверить (без ссылки на свидетельство о рождении и т.п.), что A и B во время выполнения протокола работали со своими собственными возрастами. Например, B мог работать с $j = 50$ (хотя на самом деле ему, скажем, 65 лет) и выяснить, моложе ли A пятидесяти лет или нет. С другой стороны, любое количество людей могут упорядочить свои возраста путем последовательного честного применения данного протокола.

Пример 6.2. Проиллюстрируем протокол, используя простую RSA систему ($n = 55$, $e = 7$, $d = 23$), рассмотренную ранее в начале примера 4.1. Результаты вычислений будут приводиться сразу. Предполагается, что для i и j возможны только значения 1, 2, 3, 4.

Положим вначале, что $i = 2$ и $j = 4$. Если B выбирает $x = 39$, то тогда значение $k - j$ сообщаемое A на шаге 2 равно 15. Это дает

$$y_u = D_A(15 + u), \quad 1 \leq u \leq 4 \ ,$$

и, таким образом,

$$y_1 = 26, \quad y_2 = 18, \quad y_3 = 2, \quad y_4 = 39 \ .$$

Перемешивание по модулю $p = 31$ дает

$$z_1 = 26, \quad z_2 = 12, \quad z_3 = 2, \quad z_4 = 8 \ .$$

(Это простое p не следует путать с делителем n.) После проверки условия (*) A сообщает B на шаге 4 набор

$$26,\ 18,\ 3,\ 9,\ 31 \ .$$

B замечает, что $9 \not\equiv 39 = x \pmod{31}$ и делает заключение, что $i < j$.

Если для перемешивания использовать модуль 23, то тогда

$$z_1 = 3, \quad z_2 = 18, \quad z_3 = 2, \quad z_4 = 16$$

и, таким образом, (*) не выполняется. Если бы B получил на шаге 4 набор

$$3,\ 18,\ 3,\ 17,\ 23\ ,$$

то он определил бы (кроме того, что условие (*) не было выполнено), что $i < 3$.

Положим теперь $i = j = 2$. Если B выберет $x = 48$, то $k - j = 25$ и соответственно

$$y_1 = 31, \quad y_2 = 48, \quad y_3 = 7, \quad y_4 = 24\ .$$

Если $p = 13$, то тогда B получит на шаге 4 набор

$$5,\ 9,\ 8,\ 12,\ 13$$

и после вычисления $9 \equiv 48 = x \pmod{13}$ делает вывод, что $i \geq j$.

□

Читатель может рассмотреть и некоторые другие криптосистемы с открытым ключом в качестве основы для рассмотренного протокола. Легко убедиться, что этот протокол не может быть выполнен в рамках классической криптографии, т. е. без использования односторонней функции. Действительно, участники должны передать друг другу точную информацию в таком виде, что она не может быть раскрыта из передаваемых сообщений.

6.5. Забывающая передача

Сейчас мы рассмотрим другой вариант конфиденциальной передачи данных. Участник A, владеющий неким секретом, желает передать его другому участнику B таким образом, чтобы после протокола A в действительности не знал, получил ли B секрет или нет, а B это было бы известно. Вероятность успеха такой забывающей передачи данных могла бы равняться, к примеру, $50,1\%$.

Модифицированная версия забывающей передачи данных состоит в том, что A обладает несколькими секретами. Он желает передать один из них B таким путем, чтобы только B знал, какой именно секрет был в действительности передан.

Можно привести многочисленные примеры ситуаций, когда может возникнуть необходимость в таких забывающих передачах. Например, A мог бы быть продавцом секретов, который привел список некоторых вопросов и предлагает к продаже ответ на любой из них за высокую

цену, которую мы будем предполагать одинаковой для каждого из секретов. Эти секреты могут иметь важное политическое значение, например, относиться к местонахождению какого-нибудь политического лидера. B желает купить секрет, но не хочет раскрывать, какой именно. Например, B может быть агентом сверхдержавы. Раскрытие того, что сверхдержава не осведомлена по некоторому вопросу, может оказаться делом весьма щекотливым или даже опасным.

Следует подчеркнуть, что все ограничения, возникающие в задачах частичного раскрытия секретов и забывающей передаче, легко могут быть обеспечены с помощью дополнительного участника, а именно арбитра. Вся информация передается такому арбитру, который распределяет ее среди участников согласно данным ограничениям. Значение протоколов состоит в том, что помощь честного арбитра становится ненужной. В этом и состоит основной момент, почему протоколы, основанные на криптографии с открытым ключом, открывают новые перспективы в коммуникации. Во многих случаях бывает невозможно найти какого-то арбитра, которому доверяли бы все участники. Кто мог бы быть таким арбитром между сверхдержавами?

Рассмотрим вначале случай, когда A собирается осуществить забывающую передачу какого-то секрета к B. Предположим, что секретом является разложение на множители числа n, являющегося произведением двух больших простых чисел. Отметим, что потери общности в этом случае на самом деле не происходит, поскольку секретом может быть сообщение, зашифрованное в некоторой криптосистеме RSA. Тогда знание разложения раскрывает этот секрет.

Приводимый ниже протокол является довольно простым и базируется на том факте, с которым мы уже сталкивались (например, в параграфе 6.2), что знание двух различных квадратных корней по модулю n из одного числа позволяет разложить n.

Шаг 1: B выбирает какое-то число x и сообщает A число $(x^2, \mathrm{mod} n)$.

Шаг 2: A (знающая разложение $n = pq$) вычисляет четыре квадратных корня $\pm x$, $\pm y$ из $x^2 \pmod n$ и сообщает один из них B. (Заметим, что A знает только квадрат и, таким образом, он не имеет возможности узнать, какой из квадратных корней есть x).

Шаг 3: B проверяет, будет ли квадратный корень, полученный им на шаге 2, сравним с $\pm x \pmod n$. Если да, то B не получает новой информации. В противном случае, B имеет два различных квадратных корня из одного числа по $\mathrm{mod} n$ и, следовательно, может разложить n. При этом у A нет способа узнать, какой из этих случаев имеет место.

Ясно, что вероятность для A выбрать правильное значение квадратного корня с точки зрения B равна $1/2$.

Приведем теперь другой протокол для забывающей передачи. Постановка задачи будет более общей, чем ранее. A владеет теперь двумя секретами s_0 и s_1. Она передает один из них B, но не знает, какой из них именно получит B. Предыдущая постановка получается из этой, если мы положим, что s_1 — это какая-то тривиальная информация. К тому же этот новый протокол будет *неинтерактивным*: B ничего не посылает к A.

Передача может быть осуществлена между любыми двумя пользователями A и B некоторой информационной системы. Всем пользователям этой системы известны некоторое больше простое число p, порождающий элемент g для $F^*(p)$, и некоторое число c, но никому из них не известен дискретный логарифм c. Мы предполагаем здесь, что вычисление дискретного логарифма, вообще говоря, является трудноразрешаемой задачей: невозможно вычислить $(g^{xy}, \mathrm{mod} p)$ из $(g^x, \mathrm{mod} p)$ и $(g^y, \mathrm{mod}\, p)$. Поскольку далее вся арифметика будет выполняться по модулю p, то мы будем писать, например, g^x вместо $(g^x, \mathrm{mod} p)$.

Некоторый пользователь, скажем B, вычисляет свой открытый ключ зашифрования и секретный ключ расшифрования следующим образом. B выбирает случайно бит i и число x, $0 \le x \le p-2$, и вычисляет

$$\beta_i = g^x \quad \text{и} \quad \beta_{1-i} = c(g^x)^{-1}\ .$$

Его открытым ключом зашифрования будет теперь пара (β_0, β_1), а секретным ключом расшифрования — (i, x).

Поскольку дискретный логарифм c неизвестен, то B не знает дискретных логарифмов обоих чисел β_0 и β_1. Более того, его открытый ключ шифрования не позволяет раскрыть, какой именно из этих дискретных логарифмов B в действительности знает. Прежде чем послать сообщение для B, A проверяет, что его открытый ключ зашифрования построен корректно: должно выполняться равенство $\beta_0\beta_1 = c$.

Для двоичных наборов s_1 и s_2 одинаковой длины, $s_1 \oplus s_2$ обозначает набор, полученный из s_1 и s_2 побитовым сложением по модулю 2 (без переноса). Длины любых двух двоичных наборов всегда можно сделать одинаковыми с помощью добавления нулей в начало более короткого набора.

Теперь мы готовы выполнить неинтерактивную забывающую передачу s_0 или s_1, представленных двоичными наборами.

Шаг 1: A выбирает случайные y_0 и y_1 из интервала $[0, p-2]$ и для $j = 0, 1$ вычисляет

$$\alpha_i = g^{y_j}, \quad \gamma_j = \beta_j^{y_j}, \quad r_j = s_j \oplus \gamma_j.$$

Значения α_0, α_1, r_0 и r_1 посылаются B.

Шаг 2: Используя свой секретный ключ расшифрования, B вычисляет

$$\alpha_i^x = g^{xy_i} = \beta_i^{y_i} = \gamma_i, \quad s_i = \gamma_i \oplus r_i \ .$$

Мы уже встречались ранее с пассивными и активными перехватчиками информации. В формальных определениях и доказательствах, касающихся безопасности протоколов, необходимо различать *пассивное и активное мошенничество* в протоколах. Пассивный мошенник следует протоколу, но пытается извлечь информации больше,чем на самом деле имелось в виду. Типичным представителем пассивного мошенника является описанный в параграфе 6.1 игрок в покер, который использует то обстоятельство, что все тузы есть квадратичные вычеты. Активный же мошенник может делать все, что заблагорассудится, и не следовать протоколу вовсе. Примером может быть человек, использующий неверный возраст в протоколе о сравнении возрастов. Интуитивно ясно, что в протоколе будет мало безопасности, если большая часть участников протокола будут активными мошенниками. С другой стороны, хороший протокол должен исключать возможность пассивного мошенничества.

Вернемся обратно к забывающим передачам и рассмотрим каким, образом A, обладающая несколькими секретами, может передать один из них B так, чтобы только B в итоге знал, какой из секретов был передан. Мы предполагаем принять меры только против пассивного мошенничества.

Предположим, что $s_1, \ldots, s_k$ — это секреты A, где каждое s есть двоичный набор. Таким образом, A предает гласности, например, список вопросов, а наборы s_i дают ответы на них. Протокол выполняется следующим образом.

Шаг 1: A сообщает B какую-то одностороннюю функцию f, но хранит f^{-1} только у себя.

Шаг 2: B решает купить секрет s_i. Он выбирает k случайных значений $x_1, \ldots, x_k$ из области определения f и сообщает A набор $(y_1, \ldots, y_k)$, где

$$y_j = \begin{cases} x_j, & \text{если } j \neq i \ , \\ f(x_j), & \text{если } j = i \ . \end{cases}$$

Шаг 3: A вычисляет $z_j = f^{-1}(y_j)$, $j = 1, \ldots, k$, и сообщает B числа $z_j \oplus s_j$, где используется двоичная запись числа z_j, $j = 1, \ldots, k$.

Шаг 4: B знает $z_i = f^{-1}(f(x_i)) = x_i$ и, следовательно, может вычислить s_i.

Заметим, что у B нет никакой информации о z_j при $j \neq i$, и, следовательно, он не в состоянии вычислить никакое s_j, $j \neq i$. Для A

же нет никакой возможности выделить особое значение y_j. Это относится к пассивному мошенничеству. Естественно, если B мошенничает активно и отходит от протокола, то он может узнать большее количество секретов путем представления нескольких чисел y_j в виде $f(x_j)$.

Вышеприведенный протокол использует абстрактную одностороннюю функцию f. Вместо этого мы можем полагать, что секреты шифруются с использованием RSA, причем каждый секрет шифруется с помощью своей криптосистемы RSA. Таким образом, раскрытие секрета равносильно указанию разложения на множители соответствующего модуля. Возникающий протокол базируется на той же идее, что и протокол для игры в покер из параграфа 6.2. Итак, A заранее передал гласности список секретов с указанием, о чем эти k секретов.

Шаг 1: A строит k таких RSA-систем, что в каждой из них два простых числа p_i и q_i сравнимы с 3 $\pmod 4$. (Это гарантирует, что два различных квадратных корня по модулю $n_j = p_j q_j$ из одного числа имеют различные символы Якоби.) A сообщает B ключи зашифрования (e_j, n_j), $j = 1, \ldots, k$, а также сами секреты в зашифрованном виде:

$$(s_j^{e_j}, \mathrm{mod} n_j), \quad j = 1, \ldots, k \ .$$

Числовое кодирование и последовательное деление на блоки согласуются заранее.

Шаг 2: B выбирает k чисел $x_1, \ldots, x_k$, вычисляет символы Якоби $\left(\frac{x_j}{n_j}\right)$ и квадраты $(x_j^2, \mathrm{mod} n_j)$, $j = 1, \ldots, k$. Он сообщает A эти квадраты и соответствующие символы Якоби за одним исключением. Если B собрался купить секрет s_i, то он сообщает A $(x_i^2, \mathrm{mod} n_i)$ и $-\left(\frac{x_i}{n_i}\right)$.

Шаг 3: Во всех k случаях A вычисляет квадратные корни и сообщает B квадратные корни чисел, чьи символы Якоби были получены на шаге 2.

Шаг 4: B имеет сейчас два различных квадратных корня из $x_i^2 \pmod{n_i}$ и, следовательно, может разложить n_i, а затем найти экспоненту расшифрования d_i и сам секрет s_i. Для индексов $j \neq i$ B не получил на шаге 3 никакой новой информации, поскольку ему возвратили только те квадратные корни, которые он уже знал.

Для A нет способа определить выделенный индекс i. В предположении безопасности системы RSA приведенные выше замечания относительно пассивного мошенничества относятся также и к этому случаю. B может активно мошенничать на шаге 2, выбирая несколько выделенных значений индексов i.

Пример 6.3. Вернемся к примеру 4.1. Предположим, что A хочет осуществить забывающую передачу разложения на множители числа

$n = 2773$. Пусть на шаге 1 протокола B сообщает A число 2562. Тогда A вычисляет четыре квадратных корня следующим образом. Поскольку A известны делители 47 и 59 числа 2773, то она вычисляет числа

$$(2562, \mathrm{mod}47) = 24 \text{ и } (2562, \mathrm{mod}59) = 25.$$

Квадратные корни из 24 по модулю 47 есть ± 27. А квадратные корни из 25 по модулю 59 есть ± 5. Так как обратный элемент к 59 (mod 47) равен 4, а обратный к 47 (mod 59) равен 54, то китайская теорема об остатках приводит к четырем квадратным корням $\pm 27 \cdot 59 \cdot 4$, $\pm 5 \cdot 47 \cdot 54$, или к 349, 772, 2001 и 2424 после сведения по модулю 2773. Если у B вначале было число 2001, то он, при возврате от A чисел 349 или 2424, получит в шаге 2 решающую новую информацию. Тогда как при возврате от A чисел 772 или 2001 он не получает ничего нового.

Предположим далее, что A продает секреты, зашифрованные в системах RSA, как было пояснено в последнем вышеприведенном протоколе. Пусть используемый при зашифровании некоторого секрета s модуль есть $n = 2773$ и что B на шаге 2 выбрал число 2001 и вычислил его квадрат 2562. B может вычислить символ Якоби без факторизации n:

$$\left(\frac{2001}{2773}\right) = \left(\frac{2773}{2001}\right) = \left(\frac{772}{2001}\right) = \left(\frac{193}{2001}\right) = \left(\frac{2001}{193}\right) = \left(\frac{71}{193}\right) =$$

$$= \left(\frac{193}{71}\right) = \left(\frac{51}{71}\right) = -\left(\frac{71}{51}\right) = -\left(\frac{20}{51}\right) = -\left(\frac{5}{51}\right) =$$

$$= -\left(\frac{51}{5}\right) = -\left(\frac{1}{5}\right) = -1 \ .$$

Если B желает купить s, он сообщает A пару $(2562, 1)$ и получает от нее обратно на шаге 3 или число 349, или число 2424. Поскольку 47 (но не 59) делит одновременно $349 + 2001$ и $2424 - 2001$, то B в состоянии разложить n. Если B не желает покупать s, он сообщает A пару $(2562, -1)$ и не получает на шаге 3 никакой новой информации.

Следующее замечание относится к очень простой RSA-системе с модулем $n = 55$, уже встречавшейся в начале примера 4.1. Предположим, что на шаге 2 B выбрал число 2, для которого $(2/55) = 1$. Если B желает купить соответствующий секрет, он сообщает A пару (4,-1). Тогда на шаге 3 он может получить обратно число 53, которое не дает ему ничего нового. С другой стороны, если он решил не покупать секрет и сообщает A пару (4,1), то он тем не менее получает обратно число 13, которое позволяет ему разложить число n на множители. Причина такой путаницы состоит в том, что делители n не сравнимы с 3 по модулю 4, как это должно быть согласно протоколу.

□

Вернемся к секретной продаже секретов, где теперь мы будем также допускать активное мошенничество. Ясно, что ситуация будет совершенно неуправляемой, если оба участника, A и B, будут активно мошенничать. (Во всех протоколах такого типа обычно принято предполагать, что самое большее половина из участников являются активными мошенниками.) Предположим, что B — возможный активный мошенник. Чтобы предотвратить активное мошенничество, протокол по существу модифицируется следующим образом. B связывает себя конкретным действием, в частности конкретным секретом, который он собирается купить. Это его обязательство посредством некоторой односторонней функции "запирается в ящик", но в течение протокола B должен убедить A, что он действует в соответствии с принятым обязательством. Все это он должен делать без раскрытия информации о самом действии — все это типичный случай минимального раскрытия или доказательства с нулевым знанием.

Последнее будет обсуждаться в параграфах 6.7 и 6.8. Приведем теперь протокол в очень упрощенном виде, но, однако, делающем активное мошенничество весьма маловероятным. Понятие подбрасывания числа используется в том же смысле, что и в параграфе 6.2. Как и ранее, $s_1, \ldots, s_k$ обозначают секреты, которыми владеет A.

Шаг 1: A подбрасывает к B k чисел $x_1, \ldots, x_k$. Количество двоичных разрядов в этих числах оговаривается заранее. Можно предполагать, что оно совпадает с числом разрядов y секретов s.

Шаг 2: A сообщает B некоторую одностороннюю функцию f, но оставляет у себя обратную функцию f^{-1}.

Шаг 3: Если B решил купить секрет s_i, то он вычисляет $f(x_1)$. Некоторые разряды в $f(x_i)$ совпадают с соответствующими разрядами в x_i. Пусть это будут разряды $u_1, \ldots, u_t$, если считать от начала.

Шаг 4: B сообщает A разряды всех x_α в местах u_β, для всех $1 \leq \alpha \leq k$ и $1 \leq \beta \leq t$. Он делает это таким образом, чтобы A могла бы проверить эту информацию. Например, если выполнялся протокол для подбрасывания числа из параграфа 6.2, то B сообщает для соответствующих разрядов свои исходные квадратные корни.

Шаг 5: B сообщает A набор чисел $(y_1, \ldots, y_k)$, где

$$y_j = \begin{cases} x_j, & \text{если } j \neq i \\ f(x_i), & \text{если } j = i \ . \end{cases}$$

A проверяет, что эта информация согласуется с шагом 4.

Шаг 6: A сообщает B числа $f^{-1}(y_i) \oplus s_j$, $j = 1, \ldots, k$.

Приведенный протокол основан на том обстоятельстве, что число t на шаге s равно приблизительно половине разрядов в x_i. (Предполагается, что функция f имеет более или менее случайное поведение.) Если бы B выбрал два выделенных y на шаге 5, то число "устойчивых" разрядов было бы слишком малым, поскольку B уже связал себя с числами x, порожденными на шаге 1 и должен подтвердить свое обязательство позже.

Интересно, что A может здесь пассивно смошенничать, вычисляя для каждого i те разряды, где совпадают y_j и $f^{-1}(y_j)$. Для $j = i$ разряды будут теми же, которые были переданы A на шаге 4, в то время, как для $j \neq i$ эти разряды будут различаться с очень высокой вероятностью.

Простой способ преодолеть эту трудность — это рассматривать двух покупателей B и C. Как и ранее, A обладает секретами $s_1, \ldots, s_k$. Участники A, B и C не образуют коалиции.

Шаг 1: A сообщает B и C индивидуальные односторонние функции f и g, но держит обратные функции в секрете.

Шаг 2: B сообщает C (соответственно C сообщает B) k случайных чисел $x_1, \ldots, x_k$ ($x'_1, \ldots, x'_k$ соответственно). Эти числа не надо подбрасывать, и число разрядов в них такое же, как у секретов s.

Шаг 3: B (соответственно C) вычисляет $f(x'_b)$ (соответственно $g(x_c)$) для своего выбранного индекса b (соответственно c). Значение функции и значение аргумента сравниваются для нахождения "неподвижных точек", т. е. битов, остающихся инвариантными при отображении x'_b в $f(x'_b)$ (соответственно x_c в $g(x_c)$).

Шаг 4: B сообщает C (C сообщает B) индексы неподвижных точек. Назовем эти индексы стабильными для B (соответственно для C).

Шаг 5: B (соответственно C) сообщает A числа $y_1, \ldots, y_k$ (соответственно $y'_1, \ldots, y'_k$), где все y получаются из соответствующих x изменением каждого бита, индекс которого не входит в стабильное множество, на противоположное значение.

Шаг 6: A сообщает B (соответственно C) числа $f^{-1}(y'_j) \oplus s_j$ (соответственно $g^{-1}(y_j) \oplus s_j$), $j = 1, \ldots, k$.

Ясно, что B и C получат секрет, который они хотят. Это следует из того, что $y'_b = f(x'_b)$ и $y_c = g(x_i)$. A ничего не узнает о выборе B и C, а также ни B, ни C не узнают ничего о выборе другого, поскольку

им известна только своя односторонняя функция. Попытки выбрать более чем один секрет, провалятся с очень большой вероятностью из-за шага 5, при условии, что число разрядов в записи секретов не слишком мало.

В другом простом протоколе секретной продажи секретов оба и A, и B используют свою собственную криптосистему. Эти системы могут быть и классическими, но в них должны коммутировать операторы зашифрования и расшифрования.

Шаг 1: B сообщает A случайную двоичную последовательность $y_1, \ldots, y_k$ (той же длины, что и секреты s_i).

Шаг 2: A возвращает B двоичную последовательность $z_j = E_A(s_j \oplus y_j)$, $j = 1, \ldots, k$.

Шаг 3: B, выбрав i-й секрет и зная упорядочение z_i-х, сообщает A двоичную последовательность $x = E_B(z_i)$.

Шаг 4: A передает B двоичную последовательность $D_A(x)$.

Шаг 5: B вычисляет $D_B D_A(x) = s_i \oplus x_i$, откуда он, зная y_i, находит s_i.

Возможный способ мошенничества для B состоит в выборе некоторой комбинации нескольких z вместо одного z_i на шаге 3; таким образом, он может попытаться выяснить что-нибудь о нескольких секретах сразу. Возможность успеха зависит от алгоритма зашифрования E_A.

Приведем модификацию забывающей передачи, часто называемой *совместной забывающей передачей*. A и B обладают секретами a и b соответственно, и g — предварительно выбранная функция. Выполняя протокол, B вычисляет $g(a, b)$, при этом A не знает, что вычислил B. Другими словами, A рассеянно передает указанную комбинацию своего секрета к B.

6.6. Приложения: банки и выборы

Некоторые из протоколов, рассмотренных в этой главе, кажутся искусственными или созданными для редко встречающихся целей. Однако похожие протоколы требуются и в некоторой степени уже используются во многих ситуациях. Несколько примеров будут даны в этом параграфе. Выбор каких-то конкретных протоколов всегда является некоторым компромиссом между различными соображениями безопасности и сложностью выполнения протокола.

В настоящее время в связи с компьютеризацией коренные изменения происходят в системах безналичных платежей. Возникают новые

задачи в связи с возможностью осуществления платежей через домашний компьютер. В большинстве случаев имеющиеся системы платежей являются совершенно неприемлемыми, поскольку банки и даже производители компьютеров и систем легко могут проследить, кто платит, какие суммы, кому и когда. Возникает необходимость создания систем платежей, гарантирующих защиту от мошенничества и, кроме того, обеспечивающих "ненаблюдаемость" клиентов. Только одних юридических мер здесь мало, поскольку злоупотребления могут быть не обнаружены.

Например, следующие требования относятся к условию "ненаблюдаемости" клиентов. Каждый платеж должен быть секретным для перехватившего сообщение. Если только клиент не пожелает обратного, то каждое его действие не должно связываться в одну цепь с его предыдущими действиями. Клиент должен иметь возможность действовать анонимно: банки или его партнеры не должны иметь возможность идентифицировать его. Также может потребоваться возможность оплаты произвольным получателям, причем последние могли бы проверить платежи без использования компьютерной сети.

Поскольку арбитр, которому бы все доверяли, невозможен в больших системах, то такие требования приводят к задачам построения протоколов, аналогичных рассмотренным в этой главе. Мы не будем здесь касаться деталей. Читатель, интересующийся общей моделью "ненаблюдаемых" платежных систем, может обратиться, например, к [BuP].

Криптографические протоколы могут быть также использованы при построении устройств, посредством которых люди выражают свое мнение. Такие устройства, нацеленные на обеспечение тайны, имеют особое значение для электронных выборов, т. е. выборов, проводимых с помощью компьютерной сети.

В системах тайного голосования передача сообщений должна быть защищена от перехвата. Более того, в некоторых случаях необходимо решить и проблему опознавания. Предположим, что эти требования уже выполнены, и сосредоточимся на специфических вопросах, связанных непосредственно с голосованием. В частности, рассмотрим следующие четыре условия. (1) Только законные избиратели могут иметь голос. (2) Бюллетени должны храниться в тайне. (3) Никто не имеет более одного голоса. (4) Каждый избиратель должен иметь возможность убедиться, что его голос был верно учтен при подведении итогов голосования. Удовлетворяющий (1)–(4) протокол эффективен, по крайней мере относительно очевидных форм нарушений при выборах.

Протокол непосредственно мог бы базироваться на некотором агентстве, которое проверяет легитимность каждого избирателя, подводит и

обнародует итоги голосования. Полагаем даже, что каждый избиратель посылает некоторый секретный идентификатор вместе со своим голосом, а итоги голосования публикуются в виде списка множеств

$$(*) \qquad R_1, R_2, \ldots, R_k \ ,$$

где R_i, $1 \le i \le k$, — множество, состоящее из секретных идентификаторов тех избирателей, кто голосовал за i-го кандидата или, более общо, принял i-ю избирательную стратегию. Тогда условия (1)–(4) выполняются, за исключением того, что нарушается (4) в том смысле, что наше агентство знает, как голосовал каждый избиратель. Это нарушение можно преодолеть, если у нас будут два агентства: одно для проверки легитимности (L), а другое для подведения итогов и их опубликования (C). Агентство L посылает агентству C множество N всех идентификаторов избирателей, и далее нет никаких контактов между агентствами. Теперь протокол для избирателя A выглядит следующим образом.

Шаг 1: A посылает некоторое сообщение L, например, “здравствуйте, я A”.

Шаг 2: Если A разрешено голосовать, то L посылает A некоторый идентификатор $i(A)$ к A, кроме того, вычеркивает A из списка всех избирателей. Если A не имеет права голосовать, то L посылает A отказ.

Шаг 3: A выбирает секретный идентификатор $s(A)$ и посылает C тройку $(i(A), v(A), s(A))$, где $v(A)$ есть его голос.

Шаг 4: C выясняет, находится ли число $i(A)$ в списке N. Если да, то C вычеркивает $i(A)$ из N и добавляет $s(A)$ к множеству тех избирателей, которые проголосовали за $v(A)$. Если нет, то C ничего не делает.

Шаг 5: В обусловленное заранее время C подводит и распространяет по сети итоги голосования, а также список (*).

Для увеличения безопасности можно использовать на шагах 1–3 какие-нибудь криптосистемы с открытым ключом. Сообщения будут посылаться в аутентичном, т. е. позволяющем установить подлинность сообщения и зашифрованном виде с использованием открытого ключа зашифрования получателя.

Не являющийся легитимным избирателем B может попытаться схитрить, пробуя угадать идентификатор $i(B)$. Аналогично, легитимный избиратель A может также попытаться схитрить, угадывая другие идентификационные номера. Такие попытки вряд ли пройдут, если распределить идентификационные номера среди всех обозримых чисел,

скажем, 10^6 идентификаторов распределяются среди 10^{100} первых натуральных чисел. Если же идентификаторы определяются как числа вида $10n + i_n$, $n = 1, 2, \ldots$, где i_n — n-я десятичная цифра в записи π, то тогда, конечно, они рассеяны недостаточно хорошо.

Приведенный протокол уязвим в случае возможного сговора двух агентов L и C. Ясно, что объединение информации агентов L и C раскрывает, как голосовал каждый избиратель. Чтобы преодолеть эту проблему, необходим гораздо более утонченный протокол. Такой протокол опирается на тайную продажу секретов из параграфа 6.5. Поскольку агентства сотрудничают, будем считать, что имеется одно агентство C, которое заменяет и агентство L в вышеприведенном протоколе. Кроме того, единственное отличие в протоколах состоит в том, что на шаге 2 законный избиратель A "покупает" секретно у C идентификационный номер. Это означает, что всевозможные идентификаторы открываются агентством C в зашифрованном виде. Затем C расшифровывает один из них для A, но не знает, какой именно. Вероятность того, что два избирателя могут случайно "купить" один и тот же номер, может быть сделана сколь угодно малой, если вариантов выбора зашифрованных номеров гораздо больше числа избирателей. С другой стороны, даже в зашифрованном виде идентификаторы следует "разбросать" среди чисел возможных угадываний избирателей.

В заключение следует добавить, что если использовать идеи параграфа 6.4, то агентство C не будет нужно совсем.

6.7. Убедительные доказательства без деталей

В оставшейся части этой главы мы сосредоточим внимание на следующей животрепещущей и необычной проблеме. Предположим, что P (Петр, "доказывающий") владеет некоторой информацией. Это могло бы быть доказательство какой-нибудь долго неподдающейся решению гипотезы (типа великой теоремы Ферма), разложение большого числа на простые множители, раскраска графа с помощью трех красок, некоторый пароль или идентификатор. Существенно, что информация P проверяема: существует проверяющий ее алгоритм. В отношении теорем это означает, что доказательство дано в некоторой формальной системе, в которой каждый шаг доказательства может быть проверен. P хотел бы убедить V (Вера, "проверяющая"), что, вне всякого сомнения, он владеет этой информацией.

В принципе P мог бы просто раскрыть свою информацию, так что V

могла бы выполнить проверку сама. Если информация состоит из простых делителей p и q большего числа n, P мог бы сообщать V числа p и q, и V убедилась бы, что $n = pq$. Это есть доказательство *с максимальным раскрытием*, когда V на самом деле узнает всю информацию и может в дальнейшем передать ее кому-нибудь еще и даже утверждать, что это именно она сама разложила число n на множители.

При доказательстве *с минимальным раскрытием* P убеждает V, что он владеет некоторой информацией, но это происходит таким образом, что ни один бит информации не раскрывается и, следовательно, никоим образом не помогает V определить саму информацию. V почти уверена (поскольку вероятность обмана со стороны P может быть сделана произвольно малой), что P имеет информацию, скажем, о двух делителях n. Но при этом V ничего не известно о самих делителях и она не сможет сообщить ничего о них третьей стороне.

Приведем один очень простой пример доказательства с минимальным раскрытием информации о делителях числа n:

Шаг 1: V выбирает некоторое случайное число x и сообщает P число $(x^4, \mathrm{mod} n)$.

Шаг 2: P сообщает значение $(x^2, \mathrm{mod} n)$ V.

V не получает новой информации, поскольку она сама может возвести x в квадрат. С другой стороны, известно, что извлечение квадратных корней эквивалентно разложению n на множители. На шаге 2 P извлекает не просто квадратный корень из x^4, а именно тот из них, который будет квадратным вычетом по модулю n. Определение, является ли число квадратичным вычетом или нет, также есть труднорешаемая задача, если неизвестно разложение n. Конечно, возможность успешного выполнения действий для P без знания делителей и может быть сделана еще меньше с помощью повторения протокола.

Повторим наши основные требования. Мы предполагаем, что информацией будет доказательство теоремы.

1. Доказывающий, вероятно, не сможет обмануть проверяющую. Если доказывающий не знает доказательства теоремы, то его шансы убедить проверяющую в том, что он знает доказательство, ничтожно малы.
2. Проверяющая не может перехитрить доказывающего. Она не получает из доказательства ни малейшей подсказки, кроме, конечно, самого факта, что доказывающий знает доказательство. В частности, проверяющая не сможет доказать эту же теорему кому-нибудь еще, не доказав ее сначала самостоятельно.

Протоколы, удовлетворяющие (1) и (2), противоречат распространенному убеждению, что когда мы убеждаемся, что теорема верна, обязательно происходит дополнительное проникновение в суть теоремы. Доказательства же с минимальным раскрытием не приводят к такому пониманию. Все то, что бы мы ни узнали из доказательства, можно узнать из формулировки теоремы.

Доказательства с минимальным раскрытием достигаются даже в случае, когда у доказывающего нет определенного доказательства, с которого можно было бы начать, а только некоторый аргумент, который, весьма вероятно, является верным. Например, P может найти числа p и q с помощью одного из тестов на простоту из параграфа 4.3 и вполне убедиться, что $n = pq$ есть разложение на простые множители, хотя возможно, что p или q может быть разложимо. P может теперь передать V свою убежденность с минимальным раскрытием, что означает, что V не сможет убедить какую-то третью сторону.

Вышеприведенный протокол был построен для конкретной задачи и основывался на взаимосвязи между разложением и извлечением квадратных корней. В случае если нужно построить протоколы, удовлетворяющие (1) и (2) для большого класса, такого, как задачи из класса NP, то необходимы некоторые общие идеи. Такой основной идеей в конструкции будет идея *запирающегося ящика*. Проверяющая не может открыть его, поскольку ключ находится у доказывающего. С другой стороны, доказывающий *связал себя* содержимым этого ящика в том смысле, что он не может изменить его содержимое, когда он открывает ящик. На самом деле проверяющая может наблюдать за происходящим, когда доказывающий будет открывать ящик.

Сейчас мы не будем обсуждать конструкцию этих ящиков, а вернемся к ней позже в этом же параграфе. В принципе устройство ящиков можно пояснить в терминах криптографии с открытым ключом. Запирание информации в ящик означает применение односторонней функции. Доказывающему известна обратная функция и он применяет ее, когда открывает ящик. Его привязка к содержимому ящика может быть проверена с помощью применения односторонней функции к исходной информации.

Поскольку здесь используется криптография с открытым ключом, то необходимо сделать некоторые допущения. Так, если в конструкции ящиков используется система RSA или дискретные логарифмы, то предполагается трудновыполнимым разложение на множители или вычисление дискретных логарифмов. Поскольку в большинстве случаев возможно заменить используемую криптосистему с открытым ключом, то достаточно *предполагать существование односторонней функции*.

Ящики используются в следующем доказательстве с минимальным раскрытием *3-раскрашиваемости графа*. *3-раскрашивание* графа — это сопоставление вершинам графа трех цветов B(*blue*), R(*red*) и W(*white*) таким образом, чтобы никакие две смежные (т. е. связанные ребром) вершины не получили бы один и тот же цвет. Известно, что 3-раскрашиваемость графа является NP-полной проблемой.

Итак, P желает убедить V, что он знает некоторое 3-раскрашивание графа G с t вершинами $1, 2, \ldots, t$. Протокол выполняется k этапов. Каждый этап состоит из 4 шагов и выполняется следующим образом.

Шаг 1: P подготавливает и представляет V следующие запирающиеся ящики B_i, B_i^C, $1 \le i \le 3t$ и $B_{i,j}$, $1 \le i < j \le 3t$. Каждый из ящиков B_i содержит одну из вершин и каждый из ящиков B_i^C — один из цветов, так что для каждой пары (i_1, c), где $1 \le i_1 \le t$ и $c = B, R, W$, найдется такой номер i, что вершина i_1 находится в B_i, а c в B_i^C. Более того, пары (i_1, c) появляются в парах ящиков (B_i, B_i^C) в случайном порядке.

Каждый из ящиков $B_{i,j}$ содержит 0 или 1. В ящике находится 1 тогда и только тогда, когда следующие два условия выполняются одновременно. Если вершины i_1 и j_1 оказываются в ящиках B_i и B_j соответственно, то тогда между i_1 и j_1 в графе G есть ребро. Кроме того, в 3-раскраске доказывающего цвет раскраски вершины i_1 (соответственно j_1) содержится в ящике B_i^C (соответственно B_j^C). Ящики B_i, B_i^C и $B_{i,j}$ будем называть ящиками вершин, цветов и ребер соответственно.

Шаг 2: V подбрасывает монету и сообщает P результат.

Шаг 3: (a) Если результат “орел”, то P открывает все ящики вершин и ребер. (b) Если результат “решка”, P открывает все ящики цветов и такие ящики ребер $B_{i,j}$, что вершины из ящиков B_i и B_j раскрашены доказывающим одинаково в его исходном 3-раскрашивании графа.

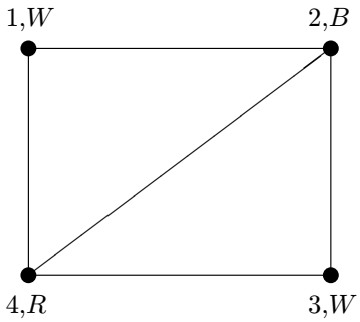
Шаг 4: (a) В этом случае V проверяет, что она получила копию графа G и $2t$ изолированных вершин. Если это так, она принимает это, в противном случае — отвергает. Сама проверка выполняется легко, поскольку открытые ящики вершин дают номера вершин, так что никаких проблем, касающихся изоморфизма графов, не возникает. (b) V проверяет, что все из $3t(t-1)/2$ ящиков ребер содержат 0, и что каждый из трех цветов появляется по t раз в ящиках цветов. Если все это так, она принимает, в противном случае — отказывается.

Следующие замечания теперь очевидны. Ящики должны быть перестроены для каждого нового этапа. В противном случае V все узнает через два этапа, выбрав варианты (a) и (b). В случае перестройки ящиков V не узнает ничего о раскраске. В варианте (a) она получает просто

копию графа G. В случае (b) она узнает только, что в 3-раскраске доказывающего никакая пара соседних вершин не окрашена одинаково и, таким образом, 3-раскраска доказывающего — правильна. P всегда проходит тест, если знает 3-раскраску графа G. В противном случае он может пытаться хитрить двумя способами. (i) Он запирает в ящики описание не графа G, а некоторого другого графа, чью 3-раскраску он знает. В этом случае он будет пойман в варианте (a). (ii) P использует некорректную 3-раскраску. Тогда он будет пойман в варианте (b). Таким образом, вероятность для лжедоказывающего пройти k этапов равна 2^{-k} и мы установили следующий результат.

Теорема 6.2 *Приведенный протокол для 3-раскрашивания удовлетворяет условиям (1) и (2). Для условия (1) вероятность обмана доказывающим равна 2^{-k}, где k — число этапов протокола.*

Пример 6.4. Пусть $t = 4$, граф G изображен ниже. Доказывающему P известна следующая 3-раскраска G:



Он готовит следующие ящики вершин и цветов:

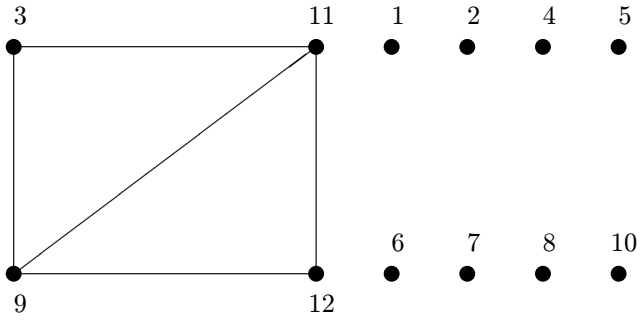
$B_1 : 2$	$B_1^C : W$
$B_2 : 1$	$B_2^C : R$
$B_3 : 1$	$B_3^C : W$
$B_4 : 1$	$B_4^C : B$
$B_5 : 1$	$B_5^C : R$
$B_6 : 1$	$B_6^C : W$
$B_7 : 1$	$B_7^C : B$
$B_8 : 1$	$B_8^C : R$
$B_9 : 1$	$B_9^C : R$
$B_10 : 1$	$B_10^C : B$
$B_11 : 1$	$B_11^C : B$
$B_12 : 1$	$B_12^C : W$

Кроме того, P приготавливает 66 ящиков ребер $B_{i,j}$ так, что 5 ящиков

$$B_{3,11}, \quad B_{3,9}, \quad B_{9,11}, \quad B_{9,12}, \quad B_{11,12}$$

содержат 1, а остальные 61 ящиков содержат 0.

В случае (a) V получает граф



где мы используем индексы ящиков вершин как метки вершин. Открытые ящики вершин информируют V, что метки $3, 11, 12, 9$ являются в том же порядке вершинами $1, 2, 3, 4$. Итак, V получает исходный граф G без раскраски и 8 изолированных вершин.

В случае (b) 18 ящиков ребер открытых для V будут:

$B_{1,3}$,	$B_{1,6}$,	$B_{1,12}$,	$B_{2,5}$,	$B_{2,8}$	$B_{2,9}$
$B_{3,6}$,	$B_{3,12}$,	$B_{4,7}$,	$B_{4,10}$,	$B_{4,11}$	$B_{5,8}$
$B_{5,9}$,	$B_{6,12}$,	$B_{7,10}$,	$B_{7,11}$,	$B_{8,9}$	$B_{10,11}$

Все эти ящики содержат 0, как и должно быть.

□

Следующий протокол будет необычным в том смысле, что запирающиеся ящики не используются, хотя и будут явно присутствовать. Итак, P желает убедить V, что он знает изоморфизм g между двумя данными графами G_1 и G_2. (По определению изоморфизм между G_1 и G_2 есть взаимно-однозначное отображение π вершин G_1 на вершины G_2, которое сохраняет соответствующие ребра: т. е. любые вершины x и y смежны в G_1 тогда и только тогда, когда $\pi(x)$ и $\pi(y)$ смежны в G_2.) Протокол состоит из k этапов со следующими тремя шагами.

Шаг 1: P порождает и сообщает V случайную изоморфную копию G_α графа G_1.

Шаг 2: V запрашивает P сообщать ей изоморфизм между G_α и G_β, где β случайно выбирается ею из индексов 1 и 2.

Шаг 3: P действует, как запрашивается.

Если P знает изоморфизм между G_1 и G_2, то выполнить шаг 3 для него будет всегда просто, поскольку он знает также и обратный изоморфизм между G_1 и G_2. В противном случае, P столкнется с проблемой, если $\beta = 2$. Можно подумать, что проверяющий узнает здесь что-то, если, например, $G_\alpha = G_2$ и $\beta = 1$. Дело в том, что V не получает никакой информации, которую она не могла бы получить *без доказывающего*: такую удачную копию G она могла бы породить сама!

Проблема *неизоморфизма графов* лежит в $Co\text{-}NP$, но неизвестно, принадлежит ли она классу NP. Конечно, она принадлежит классу P-SPACE. Используя следующий простой протокол, P может убедить V, что он знает, что данные графы G_0 и G_1 не изоморфны.

Шаг 1: V порождает случайную двоичную последовательность $i_1, \ldots, i_k$ и случайные графы $H_{i_1}, \ldots, H_{i_k}$, такие, что всегда граф H_{i_j} изоморфен G_{i_j}. V сообщает P последовательность графов H_{i_j}.

Шаг 2: P сообщает V последовательность $i_1, \ldots, i_k$.

Ясно, что у P нет способа определить эту двоичную последовательность, если исходные графы G_0 и G_1 изоморфны. Если это так, то вероятность для P быть пойманным на шаге 2 равна $1 - 2^{-k}$. Если же G_0 и G_1 не изоморфны и P обладает достаточными вычислительными возможностями для разрешения проблемы изоморфизма графов, то V будет убеждена.

Как следует из одного недавнего результата А. Шамира, P-SPACE состоит из проблем, допускающих такое интерактивное доказательство. Более точно, P обладает неограниченной вычислительной мощью, в то время как V работает в полиномиальное время и должна быть убеждена с произвольно высокой вероятностью. Этот результат особенно интересен в связи с тем, что решение, предложенное для некоторой проблемы из P-SPACE, не может быть, вообще говоря, проверено за полиномиальное время. Так что интерактивность образует здесь недостающее соединение.

Перейдем теперь к обсуждению возможных способов построения запирающихся ящиков. Не теряя общности, предположим, что каждый ящик содержит только один бит. Если первоначально предполагается содержать там больше информации, то один ящик может быть заменен несколькими, открывающимися одновременно. Описываемый ниже метод опирается на предположение, что вычисление дискретных логарифмов (по модулю p) является труднорешаемой задачей.

Вначале открываются некоторое большое простое p и порождающий элемент g группы $F^*(p)$. Это означает, что или P и V договариваются о p и g, или, в более общем случае, что p и g могут быть использованы всеми участниками, желающими заняться доказательством с минимальным раскрытием. В случае каких-либо сомнений в том, что p действительно простое или что g — порождающий элемент, можно также предполагать, что известно разложение числа $p-1$, откуда факты относительно p и g могут быть немедленно проверены.

Вначале V выбирает и сообщает P некоторое случайное число r, $1 < r < p-1$. P не может вычислить дискретный логарифм $r \pmod p$, т. е. такое число e, что $g^e \equiv r \pmod p$. Это следует из нашего допущения относительно практической невозможности вычисления дискретных логарифмов (и что не будет существенно проще, даже если известно разложение $p-1$).

Для того чтобы закрыть некоторый бит b в ящик, P выбирает случайное и секретное число y и сообщает V о "ящике": $x = (r^b g^y, \mathrm{mod} p)$. Ясно, что любой элемент из $F^*(p)$ может быть представлен как в виде $(g^y, \mathrm{mod} p)$, так и в виде $(rg^y \bmod p)$. Отсюда следует, что x не открывает ничего о запертом бите b. Когда P желает открыть этот ящик для V, он сообщает V "ключ", т. е. y. Это никоим образом не поможет V открыть другие ящики.

С другой стороны, этот метод вынуждает P привязать самого себя к биту b. Он не может открыть ящик и как 0, и как 1. Предположим противное: P может выбрать два числа y и y', такие, что

$$(g^y, \mathrm{mod} p) = (rg^{y'}, \mathrm{mod} p) \ ,$$

и затем позже объявить y и y' в качестве ключа к ящику, в зависимости от того, желает ли он появления в ящике 1 или 0. Но сейчас

$$r \equiv g^{y-y'} \pmod p$$

и, следовательно, P смог вычислить дискретный логарифм r, что противоречит нашему допущению. Все это и означает, что, закрывая бит b в ящик, P связывает себя с b и не может позже его заменить.

6.8. Доказательства с нулевым знанием

Мы сделаем сейчас более сильное ограничение для проверяющей. В то время как в условии (2) из предыдущего параграфа требовалось, что V не узнает ничего из доказательства P, то сейчас мы потребуем, чтобы V не узнала вообще ничего. По определению, протокол будет *с нулевым знанием* тогда и только тогда, когда выполняются условия (1) и

(2) и, кроме того, V *ничего не узнает от* P *такого, чего бы она смогла узнать сама без участия* P. Другими словами, V способна промоделировать протокол, как если бы P участвовал в нем, хотя на самом деле он в нем не участвует. В этом определении мы предполагаем существование односторонней функции (для того чтобы строить запирающиеся ящики).

Рассмотрим еще одну NP-полную проблему, а именно построение гамильтонова цикла в графе. По определению, цикл (т. е. путь с совпадающими начальной и конечной вершинами) в графе G называется гамильтоновым циклом, если и только если он проходит через все вершины G в точности один раз. Доказывающий, P, желает убедить проверяющую, V, что он знает гамильтонов цикл в графе G с t вершинами $1, \ldots, t$. Протокол снова имеет k этапов. Каждый этап состоит из 4 шагов и выполняется следующим образом.

Шаг 1: P запирает t вершин графа в случайном порядке в t ящиков $B_1, \ldots, B_t$. Кроме того, P подготавливает $\binom{t}{2}$ запертых ящиков B_{ij}, $1 \leq i < j \leq t$. Ящик B_{ij} содержит число 1, если в графе G имеется ребро между вершинами, запертыми в ящиках B_i и B_j. Если между этими вершинами ребра нет, то B_{ij} содержит число 0. P отдает все ящики V.

Шаг 2: V подбрасывает монету и сообщает P результат жребия.

Шаг 3: (a) Если результат — "орел", то P открывает все ящики. (b) Если результат — "решка", P открывает t ящиков $B_{i_1 i_2}$, $B_{i_2 i_3}, \ldots$, $B_{i_t i_1}$, где индексы меняются циклически и каждый индекс появляется дважды.

Шаг 4: (a) V проверяет, что она получила копию G. Эта проверка для нее будет простой, поскольку открытые ящики B_i определяют изоморфизм, использовавшийся на шаге 1. (b) V проверяет, что открытые ящики содержат число 1.

Все, что было сказано о протоколе, касающемся 3-раскрашиваемости (до формулировки теоремы 6.2), справедливо также и здесь: вышеприведенный протокол удовлетворяет условиям (1) и (2). Покажем сейчас, что он является протоколом с нулевым знанием. Предположим, что у V имеется некоторый алгоритм A (работающий недетерминированно за полиномиальное время), извлекающий некоторую информацию из ее диалога с P. Тогда V может использовать A для извлечения той же информации *даже в отсутствие* P следующим образом.

Вначале V *играет роль* P. Она побрасывает монету и, в соответствии с результатом, или применяет некоторый изоморфизм к G и запирает полученный граф в ящики, или запирает какой-то t-цикл в ящики

и, шутки ради, закладывает некоторые числа в другие ящики, чтобы сделать общее число ящиков требуемым.

Теперь, получив ящики, V *играет роль* V. Она применяет свой алгоритм A, чтобы сделать выбор между вариантами (a) и (b). Далее она или получает ту же информацию, как и в присутствии настоящего доказывающего P, или узнает, что P — лжедоказывающий. Все это V может сделать за полиномиальное время.

Те же самые аргументы применимы также к протоколу относительно 3-раскрашиваемости. Таким образом, получен следующий результат.

Теорема 6.3 *Приведенные протоколы для 3-раскрашиваемости и для гамильтоновых циклов являются протоколами с нулевым знанием.*

Рассмотрим представленный в конце параграфа 6.7 способ запирания ящиков. Там V не могла ничего извлечь из того, как P привязывает себя к конкретному биту или как открывает ящики. Эти ящики были моделируемы в том смысле, что V могла бы сделать все это сама вообще без присутствия P. Это касается как запирания, так и отпирания ящиков. Ситуация изменится, если k этапов протокола выполняются параллельно. Мы обсудим это далее в данном параграфе.

Предположим, что P знает положительное решение некоторой проблемы из класса NP, например, решение задачи о рюкзаке. (Здесь термин "положительное" противопоставлен термину "отрицательное": решений нет. Наша техника распространяется на положительные решения. Доказательства с нулевым знанием возможны также и для отрицательных решений. Например, доказательство того, что данная задача о рюкзаке не имеет решений, может быть дано в рамках подходящего формализма.) Обе проблемы, упомянутые в теореме 6.3, являются NP-полными. Это означает, что любой пример проблемы из класса NP, как, например, задача о рюкзаке, может быть за полиномиальное время сведен к любой из них. Это сведéние может быть выполнено также и проверяющей. Данный результат фиксируется в следующей теореме.

Теорема 6.4 *Для любого положительного решения произвольной проблемы из класса NP может быть дано доказательство с нулевым знанием.*

Интересная версия протокола получается, если все k этапов протоколов из теоремы 6.3 выполняются *параллельно*. Это означает, что P подготавливает сразу k наборов запертых ящиков и V задает k вопросов, по одному на каждый набор. Предположим, что V использует эти

k наборов закрытых ящиков, чтобы сформулировать свои вопросы, например интерпретируя k наборов как k чисел, применяя одностороннюю k-местную функцию к этим числам и используя первые k битов значения функции для определения вопросов. В таком случае в принципе возможно, что хотя диалог может и не содержать никакой информации о секрете P, тем не менее он не может быть восстановлен без P. Другими словами, V могла бы убедить третью сторону в факте существования секрета, хотя она не смогла бы привести никаких деталей относительно самого секрета. Фактически в этой параллельной версии V не в состоянии смоделировать k этапов за полиномиальное время.

Если бы свойство обладать нулевым знанием сохранялось даже в параллельной версии протоколов, то тогда V должна бы уметь открывать запертые ящики и как 0, и как 1. Это в точности то, что P не может делать, и такая ситуация может достигаться в случаях, когда у V есть некоторая дополнительная информация. Более конкретно, назовем запертые ящики (или метод запирания информации в ящики) *хамелеонами*, если и только если V может моделировать все то, что она увидела бы в процедуре, когда P связывает себя с битами, и, кроме того, V могла бы моделировать как процесс отпирания P ящиков как 0, так и процесс отпирания им ящиков как 1.

Ящики, основанные на дискретных логарифмах и описанные в конце параграфа 6.7, не являются хамелеонами. Если V вместо P выбирает число y, она уже не сможет открыть его для каждого их двух битов. Это означает, что протокол не может выполняться параллельно, если он должен быть с нулевым знанием. Это можно пояснить следующими аргументами. Предположим, что V дает число $(2g^e, \mathrm{mod} p) = r$ P, где e она выбрала сама. Это означает, что некий ящик, запертый P, выглядит как

$$(r^b g^y, \mathrm{mod} p) = (g^{(e+\beta)b+y}, \mathrm{mod} p) \ ,$$

где β есть дискретный логарифм 2 (mod p). Теперь V может использовать несколько таких ящиков для вычисления значения некоторой функции, чтобы определить, например, свой отклик к P. Как это может быть осуществлено без P? V могла бы, конечно, зафиксировать число y сама, но, тем не менее, чтобы открыть ящик и как 0, и как 1, она должна бы узнать β. По нашему предположению относительно дискретных логарифмов, она не знает β. И чем больше будет ящиков, тем больше будет влияния P. Так что V не сможет играть роль P. Единственный способ, когда V могла бы произвести запись протокола без P, — это когда она сама знает то, что доказывается, или еще, если она знает дискретный логарифм r. Если исключить вторую возможность,

то запись протокола может использоваться для убеждения третьей стороны b в справедливости того, что доказывается.

Добавить свойство хамелеона запирающимся ящикам возможно. Вместо случайного выбора r V выбирает случайно экспоненту e и дает P число

$$r = (g_e, \mathrm{mod} p) \ .$$

Теперь V знает дискретный логарифм r и может, в случае необходимости, убедить в этом факте P, дав доказательство с минимальным раскрытием.

Еще мы рассмотрим другую, одну из основных, NP-полную проблему, а именно проблему *выполнимости* пропозициональных формул. Эта проблема остается NP-полной, даже если мы предположим, что пропозициональные формулы находятся в *3-конъюнктивной нормальной форме*, т. е. конъюнкции дизъюнкций, где каждая дизъюнкция состоит из 3 литералов. *Литерал* — это пропозициональная переменная или ее отрицание. Например,

$$(x_1 \vee x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

$$\wedge(\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4)$$

есть пропозициональная формула в 3-конъюнктивной нормальной форме с четырьмя пропозициональными переменными и шестью дизъюнкциями. Формула *выполнима* тогда и только тогда, когда найдется такое приписывание переменным истинностных значений T (истина) и F (ложь), для которого формула принимает истинностное значение T. В нашем случае таким приписыванием будет

$$(*) \qquad x_1 = x_2 = x_3 = T, \quad x_2 = F.$$

Когда P желает убедить V, что он знает выполняющее формулу приписывание, и сделать это в стиле протоколов с нулевым знанием, то он может это сделать, следуя теореме 6.4. Мы дадим более прямой метод, напоминающий наше обсуждение относительно 3-раскрашиваемости. Такой более непосредственный подход будет более подходящим, поскольку проблема выполнимости является основной в том смысле, что задачи из класса NP могут быть сведены к ней непосредственно, см. [Sa1].

Итак, P и V известна некоторая пропозициональная формула α в 3-конъюнктивной нормальной форме. Пусть α имеет r пропозициональных переменных и t дизъюнкций. (Мы могли бы также предположить,

что α записана в алфавитном порядке, но это не важно.) P хочет убедить V, что он знает приписывание истинностных значений переменных, делающее α истинной. В качестве иллюстрации мы рассмотрим вышеприведенную формулу и приписывание (*).

Вначале P подготавливает $2r$ ящиков B_i и B_i^{TV}, $i = 1, \ldots, 2r$, называемых *ящиками переменных и ящиками истинностных значений* соответственно. Для каждой из $2r$ пар (x, y), где x есть пропозициональная переменная, а y — истинностное значение (T или F), найдется такое i, что x заперто в B_i, а y — в B_i^{TV}. Кроме того, пары (x, y) расположены в случайном порядке среди пар из ящиков (B_i, B_i^{TV}). В нашем примере будет 8 пар ящиков, например,

$$\begin{array}{ll} B_1 : x_4 & B_1^{TV} : T \\ B_2 : x_2 & B_2^{TV} : F \\ B_3 : x_1 & B_3^{TV} : F \\ B_4 : x_4 & B_4^{TV} : F \\ B_5 : x_3 & B_5^{TV} : T \\ B_6 : x_3 & B_6^{TV} : F \\ B_7 : x_1 & B_7^{TV} : T \\ B_8 : x_2 & B_8^{TV} : T \end{array}$$

Кроме того, P готовит $(4r)^3$ ящиков $B_{i,j,k}$, где три индекса меняются от 1 до $2r$ и от $\neg 1$ до $\neg 2r$ и каждый ящик содержит 0 или 1. Число 1 находится в ящике $B_{i',j',k'}$ только в случае $i' = i$ или $i' = \neg i$, $j' = j$ или $j' = \neg j$, $k' = k$ или $k' = \neg k$, α содержит дизъюнкцию, три переменных которой — это переменные из ящиков B_i, B_j, B_k (в таком порядке и с отрицанием индекса, если это указывается в индексах i', j', k') и дополнительно, если три истинностных значения, которые P приписывает эти трем переменным (в своем конкретном выполняющем приписывании), содержатся в ящиках B_i^{TV}, B_j^{TV}, B_k^{TV} (в таком порядке). Ящики $B_{i',j',k'}$ будем называть *ящиками приписывания*. Таким образом, t из них содержат число 1. В нашем примере шесть ящиков приписывания, содержащих число 1, будут такими

$$\begin{array}{lll} B_{7,2,\neg 1} & B_{2,5,\neg 1} & B_{\neg 7,2,5}\ , \\ B_{\neg 7,\neg 2,\neg 5} & B_{7,5,1} & B_{\neg 2,5,1}\ . \end{array}$$

Ящики перечислены в том же порядке, что и дизъюнкции.

Протокол теперь выполняется вполне аналогично протоколу для 3-раскрашиваемости. На каждом этапе протокола P подготавливает и передает V закрытые ящики, описанные выше. У V теперь есть две возможности.

По желанию V P открывает для нее все ящики, за исключением истинностных значений. Из ящиков приписывания, содержащих число

1, V получит исходную пропозициональную формулу α. Таким образом она узнает, что P, закрывая ящики, действительно использовал формулу α, но она не узнает никакой информации о сделанных P приписываниях истинностных значений.

V также может попросить P открыть все ящики истинностных значений. Тогда P открывает для нее и все те ящики приписывания $B_{i',j',k'}$, где каждый из индексов имеет вид x с F в ящике $B_x^T V$, или вид $\neg x$ с T в ящике $B_x^T V$. Если во всех таких ящиках содержится число 0, то тогда приписывание истинностных значений, сделанное P, будет корректным: никакая дизъюнкция, принимающая значение F при данном приписывании, не входит в α. Итак, все дизъюнкции, входящие в α, принимают значение T при сделанном P приписывании. V будет убеждена в этом, хотя о самом приписывании она не узнает ничего. В нашем примере P откроет все ящики приписывания, у которых каждый из трех индексов принадлежит множеству $2, 3, 4, 6, \neg 1, \neg 5, \neg 7, \neg 8$.

Следующий результат получается так же, как и в теореме 6.2: вероятность обмана доказывающим умножается на $1/2$ после выполнения каждого этапа.

Теорема 6.5 *Приведенный протокол для выполнимости является протоколом с нулевым знанием.*

Любая из теорем 6.3–6.5 может быть использована для превращения любого математического доказательства в доказательство с нулевым знанием. Положим, что вы знаете доказательство, скажем, последней теоремы Ферма. Предположим далее, что ваше доказательство формализовано в рамках некоторой формальной системы. Это означает, что никакого "размахивания руками" здесь нет: проверяющая может просто проверить, что каждый шаг доказательства следует из правил этой системы. Предположим, наконец, что известна верхняя оценка длины доказательства.

Доказательство может быть отыскано недетерминированной процедурой, работающей за полиномиальное время. Эта процедура вначале угадывает доказательство, а затем проверяет его корректность шаг за шагом. С другой стороны, эта процедура (или, скажем, недетерминированная машина Тьюринга) может быть описана в терминах пропозициональных формул как формула α в 3-конъюнктивной нормальной форме, причем α будет выполнима тогда и только тогда, когда теорема имеет доказательство, длина которого не превосходит данной границы. Построение α эффективно в том смысле, что любому, кому известно доказательство теоремы, известно также и приписывание переменных, выполняющее α. Таким образом, вы в состоянии убедить проверяющего, что вам известно доказательство теоремы, не раскрывая никакой

информации о самом доказательстве, за исключением верхней оценки его длины.

Осталось сделать несколько дополнительных замечаний. При получении результатов, таких, как теорема 6.4, требуется существование односторонних функций. Действительно, в наших протоколах с нулевым знанием односторонние функции используются при построении запирающихся ящиков. Это означает не что иное, как то, что доказывающий *раскрывает свой секрет* проверяющей *в зашифрованном виде*. И хотя в диалоговом режиме проверяющая не извлекает никакой информации, в принципе возможно, что позже или случайно, или сделав существенные вычислительные усилия, она сможет раскрыть криптосистему и узнать секрет целиком. Напомним, что, например, 3-раскрашивание передается проверяющей в закрытых ящиках на каждом этапе соответствующего протокола.

Мы не будем здесь касаться протоколов, называемых протоколами с *совершенным нулевым знанием*. В таких протоколах V не получает вообще никакой информации о секрете P (кроме самого факта его существования), в то время как в протоколах с нулевым знанием, которые обсуждались выше, предполагается, что V не получает никакой информации в диалоговом режиме или за полиномиальное время. Читатель может поразмышлять о смысле доказательств с нулевым знанием с запирающимися при помощи криптосистемы RSA ящиками в случае, когда доказываемая теорема — “существует линейный алгоритм факторизации”.

В рассмотренных выше протоколах вероятность обмана доказывающим убывает очень быстро с увеличением числа этапов. Однако произвольно высокую безопасность нельзя достичь за конечное число этапов. Эта техника может быть модифицирована для сочетания высокой безопасности с фиксированным числом этапов. В некоторых случаях рассматриваются даже *неинтерактивные системы доказательств с нулевым знанием*. Полученные здесь результаты [BeG] могут быть применены в следующей ситуации. P и V порождают совместно длинную случайную последовательность, а затем P отправляется в кругосветное путешествие. Как только он открывает какую-то теорему, он посылает V открытку о доказательстве своей новой теоремы с нулевым знанием. Этот процесс будет неизбежно неинтерактивным, поскольку у P нет никакого предсказуемого адреса.

6.9. Доказательства с нулевым знанием подлинности

Одна из проблем в процедурах идентификации, таких, как карточка удостоверения личности, кредитные карточки или компьютерные пароли, состоит в том, что участник P доказывает свою подлинность, раскрывая некое слово $i(P)$, которое записано в память или напечатано на карточке. Противник, сотрудничающий с нечестной проверяющей, может либо получить экземпляр самой карточки, либо узнать само слово $i(P)$. Позже этот противник может использовать $i(P)$ и притвориться, что он есть P и, таким образом, получить доступ или обслуживание, подразумеваемое $i(P)$.

Очевидное решение этой проблемы состоит в использовании доказательства с нулевым знанием и убедить проверяющую V, что P *знает* $i(P)$, при этом не раскрыв ни одного бита информации о $i(P)$. Такое доказательство идет на шаг дальше, чем те доказательства с нулевым знанием, которые рассматривались в предыдущем параграфе. Раньше P раскрывал один бит информации V, а именно то, что теорема верна, что имеется 3-раскрашивание или выполняющее формулу приписывания переменных и т. д. Сейчас же ни один бит не раскрывается. Коротко разницу можно выразить так, сказав, что если раньше мы говорили о *доказательствах с нулевым знанием теорем*, то теперь мы говорим о *доказательствах с нулевым знанием знаний*.

Конечно, последний вид доказательств может быть расширен также и до доказательства теорем. Это означает, например, что P убеждает V, что он справился с последней теоремой Ферма, не раскрывая ни одного бита своей информации, даже того, доказал ли он саму теорему или нашел контрпример! Способ сделать это — положить $i(P)$ состоящим из информации P, т. е. с утверждения теоремы или ее отрицания с последующим доказательством или контрпримером.

В приводимом протоколе предполагается существование заслуживающего доверия агентства. Единственной задачей такого агентства будет опубликование модуля n, равного произведению двух больших простых чисел p и q, и хранение самих простых чисел в секрете. По техническим причинам, которые мы поясним ниже, эти простые числа предполагаются сравнимыми с $3 \pmod 4$. Опубликовав n, агентство может прекратить свое существование.

Для P секретный идентификатор $i(P)$ состоит из k чисел $c_1, \ldots, c_k$, где $1 \leq c_j < p$. Его публичный идентификатор $p_i(P)$ состоит из k чисел $d_1, \ldots, d_k$, где $1 \leq d_j < p$, и каждое d_j удовлетворяет одному из

сравнений

$$(*) \qquad d_j c_j^2 \equiv \pm 1 \pmod{n} \ .$$

Проверяющей V известны n и $\pi(P)$. P желает убедить ее, что он знает $i(P)$. Следующие четыре шага образуют один этап протокола. Количество этапов уменьшает вероятность схитрить для P.

Шаг 1: P выбирает случайное число r, вычисляет числа $(\pm r^2, \mathrm{mod}\, n)$ и сообщает одно из них, обозначим его x, V.

Шаг 2: V выбирает подмножество S множества $\{1, \ldots, k\}$ и сообщает его P.

Шаг 3: P сообщает V число

$$y = (rT_c, \mathrm{mod} n) \ ,$$

где T_c есть произведение чисел c_j, у которых индекс j принадлежит S.

Шаг 4: V проверяет условие

$$x \equiv \pm y^2 T_d \pmod{n} \ ,$$

где T_d есть произведение чисел d_j, таких что индекс j принадлежит S. Если условие не выполняется, то V отказывает. В противном случае начинается очередной новый этап.

Заметим, что проверочное условие на шаге 4 должно выполняться, поскольку

$$y^2 T_d \equiv r^2 T_c^2 T_d \equiv \pm r^2 \equiv \pm x \pmod{n} \ ,$$

где второе сравнение есть следствие (*). Использование r необходимо, поскольку в противном случае V могла бы узнать все c_j, выбирая $S = \{j\}$. Специальный же вид простых p и q обеспечивает, что все d-числа могут пробегать все целые числа с символом Якоби равным $+1$ $(\mathrm{mod}\ n)$). Из этого следует, что V может быть уверена, что c-числа существуют. В (*) по умолчанию подразумевалось, что $(c_j, n) = 1$ для всех j. Если же это не так, то n может быть разложено и весь мир рушится! Небольшая техническая деталь, полезная на практике, состоит здесь в том, что, определяя (*), лучше искать обратные к квадратам c-чисел, а не возводить в квадрат обратные к самим c-числам. Конечно, весь протокол основан на том, что извлечение квадратных корней по модулю n, когда разложение n неизвестно, является труднорешаемой задачей.

Все это означает, что V не получает никакой информации о c-числах и на самом деле V может даже играть обе роли P и V в этом протоколе. С другой стороны, единственный способ для P схитрить состоит

в угадывании множества S заранее и предоставлении $(\pm r^2 T_d, \mathrm{mod} n)$ в качестве x на шаге 1 и $y = r$ на шаге 3. Вероятность успешного угадывания есть 2^{-k} и, следовательно, 2^{-kt} в t этапах. Причина такого быстрого убывания в том, что k чисел из идентификатора P вносят элемент параллелизма в протокол. Предполагая трудновычислимость для разложения на множители и извлечения квадратного корня по модулю n, наш протокол составляет доказательство подлинности с нулевым знанием. Отсюда следует, что даже нечестная Вера не сможет извлечь никакой информации, которая могла бы позже использоваться, чтобы убедить честную Веру в знании $i(P)$.

Отметим попутно, что для целей этого параграфа нет нужды в строгом формализме. При таком формализме P и V были бы машинами, выполняющими алгоритмы в некоторых временных границах и имеющими доступ к общим или раздельным случайным числам. При обсуждении многих тонкостей такой более глубокий формализм будет полезным.

Пример 6.5. Надежное агентство опубликовало модуль $n = 2773$. Секретный идентификатор $i(P)$ для P состоит из шестерки чисел

$$\begin{array}{lll} c_1 = 1901, & c_2 = 2114, & c_3 = 1509\ , \\ c_4 = 1400, & c_5 = 2001, & c_6 = 119\ . \end{array}$$

(См. также пример 4.1.) Квадратами этих чисел по модулю n в том же порядке будут

$$582,\ 1693,\ 448,\ 2262,\ 2562,\ 296.$$

Теперь P выбирает свой публичный идентификатор $\pi(P)$ из таких шести чисел

$$\begin{array}{lll} d_1 = 81, & d_2 = 2678, & d_3 = 1207\ , \\ d_4 = 1183, & d_5 = 2681, & d_6 = 2595\ . \end{array}$$

Сравнения (*) выполняются для $j = 1, \ldots, 6$, кроме того, $+1$ стоит в правой части для $i = 1, 3, 4, 5$ и -1 для $j = 2, 6$.

Предположим, что P выбирает $r = 1111$ и сообщает V число

$$x = (-r^2, \mathrm{mod} n) = 2437\ .$$

Предположим, что V выбирает $S = \{1, 4, 5, 6\}$ и вычисляет $T_d = 1116$. P вычисляет $T_c = 96$ и сообщает V число $y = 1282$. Поскольку

$$y^2 T_d = 1282^2 \cdot 1116 = 2437 = x \pmod{n}\ ,$$

проверочное условие выполняется.

Аналогично, выбор $r = 1990$, $x = (r^2, \mathrm{mod} n) = 256$ и $S = \{2, 3, 5\}$ дает значения

$$T_d = 688,\ T_c = 1228,\ y = 707.$$

Проверочное условие $-y^2 T_d \equiv -2517 \equiv x \pmod{n}$ выполняется.

□

Мы уже отмечали, что даже нечестная проверяющая не может извлечь никакой информации, которая могла бы впоследствии использовать для убеждения честной проверяющей в знании $i(P)$. Тем не менее в диалоговом режиме возможны варианты более хитроумного обмана. Предположим, что нечестная проверяющая и доказывающий, V_c и P_c, сотрудничают в попытке убедить честную проверяющую V, что P_c знает идентификатор $i(P)$ честного доказывающего P. Предположим далее, что V_c имеет возможность проверить знание P своего идентификатора $i(P)$. Например, P хочет заплатить деньги V по счету. Тогда в то же самое время P_c, имеющий возможность секретно общаться с V_c по радио или по телефону, пытается получить доступ в совершенно секретную зону, тот доступ, который предоставляет V, если продемонстрировано знание $i(P)$. Теперь P_c и V_c могут действовать просто как канал связи и весь протокол на самом деле будет выполняться между V и P. И V будет в итоге убеждена в знании $i(P)$, хотя получит неверное представление, что P_c знает $i(P)$!

Сейчас мы обсудим другую схему проверки удостоверения, основанную на задаче рюкзачного типа. Вначале мы приведем упрощенный вариант этой схемы, а затем дадим немного более сложный. Последний может быть обобщен и далее, хотя и не совсем еще понятно, в какой степени, если это вообще так, такое обобщение и усложнение дают вклад в безопасность всей схемы.

Пусть $A = (a_1, \ldots, a_n)$ — рюкзачный вектор с честным n. NP-полной является задача подбора половины компонент вектора A, таких, что их сумма совпадает с суммой оставшейся половины компонент. Таким образом и более общая следующая задача является NP-полной. Даны рюкзачный вектор A и вектор $B = (b_1, \ldots, b_n)$ с целыми компонентами, возможно и отрицательными. Требуется найти, если это возможно, перестановку B_p вектора B так, чтобы $AB_p = 0$. Например, если

$$A = (3, 7, 8, 2, 12, 14), \qquad B = (1, 1, 1, -1, -1, -1)\ ,$$

то подстановка p, переставляющая 2-й и 5-й компоненты и оставляющая остальные на месте, удовлетворяет условию, поскольку

$$A = (1, -1, 1, -1, 1, -1) = 0.$$

(Здесь второй вектор в произведении подразумевается как вектор-столбец.)

Установочные данные теперь таковы. Заслуживающее доверия агентство публикует рюкзачный вектор $A = (a_1, \ldots, a_n)$ (n не обязательно четно). Публичный идентификатор $\pi(P)$ для P есть $B = (b_1, \ldots, b_n)$ с целыми компонентами. Его секретный идентификатор $i(P)$ — это подстановка p, такая, что $AB_p = 0$. В протоколе, кроме того, используется *криптографическая хеш-функция* $h(x, y)$. Мы не будем определять хеш-функции формально. Для нас здесь существенно то, что значение $h(x, y)$ может быть легко вычислено по x и y, в то время как x и y не могут быть восстановлены по значению функции и, кроме того, $h(x, y)$ не длиннее по сравнению с x и y. Ранее встречавшийся оператор $\oplus$ сложения по модулю 2 есть простая хеш-функция, если только $x \oplus y$ не допускает утечки информации о x или y. Хеш-функция $h(x, y)$ публикуется агентством или согласуется между P и проверяющей V. Конечно, желательно, чтобы даже $h(x, y)$ и один из аргументов не раскрывали другой. Ясно, что это условие не выполняется для оператора $\oplus$.

Каждый этап протокола, в котором P пытается убедить V в том, что он знает $i(P)$, состоит из следующих шагов:

Шаг 1: P выбирает случайный вектор R и случайную подстановку q (и то и другое размерности n) и сообщает V значения $h(q, AR)$ и $h(pq, R_q)$.

Шаг 2: V выбирает число $d = 0$ или $d = 1$ и запрашивает у P вектор $C = R_q + d \cdot B_p q$. Получив C, V запрашивает у P или подстановку q, или подстановку pq.

Шаг 3: Если запрашивалось q, то V проверяет условие $h(q, A, C) = h(q, AR)$. Если запрашивалось pq, V проверяет условие

$$h(pq, C - dB_p q) = h(pq, R_q) \ .$$

Отметим сразу, что у V есть все данные, необходимые на шаге 3, полученные на шагах 2 и 3 или из публичной информации. Справедливость второго проверочного условия очевидна по определению C. Справедливость первого условия вытекает из равенств

$$A_q C = A_q(R_q + dB_p q) = A_q R_q + dA_q B_p q = AR \ .$$

(Равенство $A_q B_p q = 0$ выполняется, поскольку $AB_p = 0$ и, следовательно, произведение также равно 0, если оба множителя переставлялись одной и той же подстановкой.)

Возвращаясь к примеру перед протоколом, положим

$$R = (15, 1, 5, 9, 2, 6), \quad d = 1 \quad \text{и} \quad q = (1234) \ .$$

Здесь использована обычная запись подстановки: q есть отображение, которое переставляет циклически компоненты 1,2,3,4 и оставляет две другие компоненты на месте. Тогда

$$\begin{array}{lcl} R_q & = & (9,15,1,5,2,6), \quad pq = (12534)\ , \\ B_{pq} & = & (-1,1,-1,1,1,-1)\ , \\ C & = & (9,15,1,5,2,6) + (-1,1,-1,1,1,-1) = (8,16,0,6,3,5)\ , \\ A_q & = & (2,3,7,8,12,14)\ , \quad A_qC = 218 = AR\ . \end{array}$$

Итак, проверочное условие будет выполнено.

Заметим, что число подстановок огромно даже для относительно небольших значений n. Это очень важно с точки зрения безопасности.

Рис. 6.1.

В более сложном варианте этого протокола A есть $m \times n$ матрица с целыми коэффициентами, а A_p матрица, полученная из A применением подстановки p к *столбцам* A. Фиксируется небольшое простое число s, обычно $s = 251$. A и s публикуются агентством или согласуются всеми пользователями. Как и раньше, открытый идентификатор $\pi(P)$ для f — это n-вектор B. Его секретный идентификатор $i(P)$ есть подстановка p, такая, что

$$AB_p \equiv 0 \pmod{s}\ ,$$

где справа подразумевается m-вектор из нулей (в нашей ранней простой версии $m = 1$ и сравнение есть просто равенство). Сам протокол в основном тот же, что и раньше, хотя выбор q тоже будет более общим,

теперь $0 \leq d < s$. Компоненты всех векторов берутся по модулю s и, кроме того, AR и A_qC теперь m-векторы. Все остальное остается без изменений. Также и справедливость проверочных условий в точности следует, как и раньше, и, таким образом, честный доказывающий P всегда проходит тест. Легко видеть, что вероятность успеха у нечестного доказывающего P (незнающего подстановку p) в лучшем случае $(s+1)/2s$. Протокол будет с нулевым знанием, поскольку индивидуальные сообщения, посылаемые P, не несут никаких знаний. Дальнейшее обобщение этой схемы получается, например, путем замены произведений матриц на векторы на произведения матриц или даже тензоров.

Приложение A

Элементы теории сложности

Следующие два приложения представляют собой лишь краткое введение в те области теории сложности и теории чисел, которые используются в настоящей книге. Имеется множество хороших учебников как по теории сложности, так и по теории чисел.

С точки зрения классической математики проблемы в криптографии являются тривиальными в том смысле, что они могут быть разрешены за конечное число попыток. Однако сведение к конечному числу случаев не имеет особого смысла в случае, когда число самих случаев не поддается обработке на практике. И если мы не способны расшифровать некоторые сообщения в рамках некоторых временных границ, то мы можем забыть про все остальное, поскольку, когда время уйдет, ситуация может полностью измениться.

Временная сложность алгоритма есть функция от длины входа. Говорят, что алгоритм имеет временную сложность $f(n)$ тогда и только тогда, когда для всех входов *длины* n выполнение алгоритма заканчивается за не более чем $f(n)$ шагов. Если n — число, то его длиной будет число цифр или число разрядов в двоичном представлении n. Конечно, для одной и той же проблемы могут существовать как медленные, так и быстрые алгоритмы. В некоторых случаях возможно даже неограниченное ускорение. Трудной задачей является получение нижних оценок сложности, т. е. показать, например, что любой алгоритм для некоторой проблемы имеет по крайней мере квадратичную оценку сложности.

Ясно, что временная сложность зависит от *модели* алгоритмов, которую мы подразумеваем. Число шагов будет меньше, если на одном шаге

будет выполняться больше работы. Однако фундаментальные понятия, такие, как полиномиальная временная сложность, в значительной степени не зависят от выбора модели алгоритмов. Конечно, это касается только моделей, выбранных разумно. Например, подпрограмма, проверяющая простоту целых чисел, не должна включаться в один шаг алгоритма!

Чтобы быть более конкретным, мы выберем *машину Тьюринга* в качестве нашей модели алгоритмов. Машина Тьюринга работает в дискретные моменты времени. В каждый момент времени она может находиться в некотором внутреннем состоянии (памяти), число которых конечно. Считывающе-записывающая головка сканирует буквы, записанные на ленте, по одной в каждый момент времени. Каждая пара (q, a) определяет тройку (q_1, a_1, m), где q и q_1 — это состояния, a и a_1 — буквы и m означает одно из трех значений: "налево", "направо" и "на месте". Все это означает, что в состоянии q, сканируя букву a, машина переходит в состояние q_1, записывает a_1 на место a (возможно, $a_1 = a$) и передвигает головку в соответствии с m.

Если считывающе-записывающая головка может "свалиться" с ленты, т. е. когда требуется перейти налево, а машина сканирует крайний слева квадрат ленты, то к ленте добавляется слева новый пустой квадрат. То же самое делается в отношении правого конца ленты. Эта способность бесконечного расширения внешней памяти может рассматриваться как встроенная особенность устройства каждой машины Тьюринга.

Сама лента может рассматриваться и как потенциально бесконечная память, и как входной и выходной канал. Формат входа-выхода описывается следующим образом. Машина начинает свои вычисления, находясь в некотором начальном состоянии и считывая крайнюю слева букву данного входного слова. Вычисление заканчивается, когда машина достигает некоторого заключительного состояния. Тогда машина останавливается и оказавшееся на ленте слово составит выход. При считывании выхода некоторые вспомогательные буквы могут игнорироваться. Читатель отсылается к [Sa1] за более формальными определениями, а также обсуждением универсальности этой модели.

Теперь ясно, что означает один шаг вычислений. Мы можем определить *функцию временной сложности*, связанную с машиной Тьюринга A, равенством

$$f_A(n) = \max\{m | A \text{ останавливается после } m \text{ шагов на входе } w \text{ с } |w| = n\}.$$

Мы предполагаем для простоты, что A останавливается, т. е. достигает заключительного состояния для всех входов. Конечно, это не так по

отношению к произвольной машине Тьюринга. Назовем машину Тьюринга A *полиномиально ограниченной*, если существует многочлен $p(n)$, такой, что $f_A(n) \leq p(n)$ для всех n. Обозначение P используется для класса проблем, допускающих решение на полиномиально ограниченных машинах Тьюринга.

Проблема называется (вычислительно) *труднорешаемой* (иногда даже практически неосуществимой), если она не принадлежит классу P. Легкорешаемые проблемы (т. е. проблемы из P) образуют несколько подклассов в P с очевидными определениями: задачи с линейной, квадратичной, кубичной и другой временной сложностью. Неформальные понятия *легкой* проблемы означают, что степени многочленов малы, по крайней мере в указанных выше пределах.

Рассмотренная выше машина Тьюринга является *детерминированной*: внутреннее состояние и буква на ленте однозначно определяют ее поведение. Чтобы подчеркнуть, что имеется в виду детерминированная машина Тьюринга, часто говорят о *детерминированной временной сложности*.

Недетерминированная машина Тьюринга имеет несколько возможностей своего поведения при считывании буквы. Соответственно входные слова приводят к нескольким вычислениям. Это можно представить как то, что машина делает догадки или использует произвольное число параллельных процессоров. Для каждого входа w рассматривается наикратчайшее успешное вычисление $S(w)$ (т. е. вычисление, ведущее к заключительному состоянию). Функция временной сложности недетерминированной машины Тьюринга A определяется как

$$f_A(n) = \max\{1, m| \text{ в } s(w) \ m \text{ шагов для } w \text{ с } |w| = n\} \ .$$

Рассматривается пара $(1, m)$, поскольку для некоторых n, возможно, нет вообще таких входов длины n, которые приводят к заключительному состоянию.

Понятия полиномиально ограниченной недетерминированной машины Тьюринга и соответствующий класс проблем, NP, определяются в точности, как и в детерминированном случае. Проблемы из класса P являются легкорешаемыми, в то время как проблемы из NP обладают тем свойством, что легкорешаемой оказывается проверка: будет удачно угаданное решение проблемы верным или нет. Временную границу для недетерминированной машины Тьюринга можно представить себе как временное ограничение на проверку того, будет или нет угаданное решение проблемы верным. Относительно разложения на множители неизвестно, лежит ли эта проблема в классе P, хотя, конечно, она из NP: достаточно угадать сомножители и проверить догадку, вычислив их произведение.

Из определений следует, что P включается в NP, а знаменитой открытой проблемой является вопрос: совпадают эти классы или нет, $P = NP$? Несмотря на то что этот вопрос открыт, можно указать много NP-полных проблем. Проблема является NP-*полной*, если и только если она из NP и, кроме того, является NP-*трудной*, т. е. любая проблема из NP может быть сведена к ней за полиномиальное время. Отсюда вытекает, что $P = NP$ тогда и только тогда, когда некоторая NP-полная проблема принадлежит классу P. В этом случае произвольная проблема из NP может быть разрешена за полиномиальное время, поскольку сперва за полиномиальное время она может быть сведена к рассматриваемой NP-полной проблеме, которая в свою очередь по предположению может быть разрешена за полиномиальное время. Ясно, что композиция двух полиномов снова будет полиномом.

Общее убеждение состоит в том, что $P \neq NP$. Таким образом, NP-полные проблемы рассматриваются как труднорешаемые. Помимо NP, термины "трудная" и "полная" используются в том же смысле и в связи с другими классами проблем.

То, что конкретная проблема является NP-трудной, доказывается тем, что некоторая проблема, про которую уже известно, что она является NP-трудной, может быть сведена за полиномиальное время к рассматриваемой проблеме. Если мы хотим показать, что данная проблема является NP-полной, то мы должны показать также, что она из NP.

Однако нам нужно что-то, с чего можно начать: проблема, NP-полнота которой может быть установлена прямым рассуждением безо всяких сведений. Для этой же цели очень подходящей задачей является проблема *выполнимости для регулярных формул пропозиционального исчисления*, сокращенно РФПИ. Такие формулы получаются из переменных путем использования операций конъюнкции $\wedge$, дизъюнкции $\vee$ и отрицания $\neg$. Мы опускаем очевидное рекурсивное определение. *Приписывание истинностного значения* РФПИ α есть отображение множества переменных, входящих в α, во множество {истина,ложь}. Истинностное значение α может быть вычислено для любого истинностного приписывания переменных с использованием истинностных таблиц конъюнкции, дизъюнкции и отрицания. Две РФПИ *эквивалентны*, если они принимают одинаковые истинностные значения для всех истинностных приписываний переменных. РФПИ α *выполнима*, если она принимает значение "истина" хотя бы для одного истинностного приписывания переменных. Например, РФПИ

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge \neg x_3$$

не выполнима. Действительно, последняя дизъюнкция вынуждает приписывание $x_3 =$ ложь. Далее из третьей дизъюнкции получим

x_1 = ложь и из второй — x_2 = истина. Но это приписывание противоречит первой дизъюнкции. Рассмотренная РФПИ находится в *конъюнктивной нормальной форме*: конъюнкция дизъюнкций, где члены каждой дизъюнкции — это *литералы*, т. е. переменные или их отрицания. Кроме того, говорят, что РФПИ записана в 3-*конъюнктивной нормальной форме*, если каждая дизъюнкция содержит не более трех литерал.

Для проблемы выполнимости РФПИ может быть дано прямое доказательство ее NP-полноты. На самом деле тот факт, что вычисление на данной машине Тьюринга с данным входом является успешным, эквивалентно тому, что некоторая РФПИ будет выполнимой. Детали можно найти, например, в [Sa1]. Этот результат остается справедливым, если ограничиться РФПИ в 3-конъюнктивной нормальной форме. Выполнимость может быть, конечно, выяснена проверкой всех возможных истинностных приписываний переменных. Это, однако, ведет к экспоненциальной временной сложности.

Пространственная сложность определяется аналогично. Когда вход машины Тьюринга имеет длину n, то в исходной ситуации занято n ячеек ленты. В процессе вычислений могут понадобиться новые ячейки, их число и дает пространственную сложность. Полиномиальные ограничения могут быть рассмотрены и в этой ситуации. Это приводит к классам P-SPACE и NP-SPACE. Ясно, что временной класс включается в соответствующий пространственный класс, поскольку, для того чтобы увеличить ленту на одну ячейку, необходим один временной такт. Для пространственных классов можно на самом деле показать, что P-SPACE=NP-SPACE. Соответственно имеем следующую цепь включений:

$$P \subseteq NP \subseteq P\text{-SPACE} = NP\text{-SPACE} .$$

Будут ли данные два включения строгими или нет остается пока знаменитой открытой проблемой.

Класс Co-NP состоит из проблем, “дополнение” к которым лежит в NP. Например, дополнением проблемы “Является ли данное целое число простым?” будет “Является ли данное целое число составным?”. Формальное определение может быть дано, если рассматривать проблемы как языки. Ясно, что если проблема из P, то тогда также и ее дополнение из P: тот же самый алгоритм будет работать также и для дополнения. Это будет не так в недетерминированном случае. На самом деле, взаимное расположение классов NP и Co-NP неизвестно, хотя есть общепринятое впечатление, что $NP \neq Co\text{-}NP$. Легко понять, что если дополнение некоторой NP-полной проблемы лежит в NP, то тогда $NP = Co\text{-}NP$.

В приложениях теории сложности к криптографии необходимо помнить о некоторых предостережениях. При рассмотрении полиномиальной сложности безусловно важной оказывается степень полинома. Например, n^{1000} растет в конечном счете медленнее, чем $n^{\log\log n}$, но тем не менее будет, вероятно, гораздо худшей верхней оценкой изучаемых величин. В криптографии сложность в среднем более важна, чем сложность в худшем случае. Предположим, что некоторый пользователь выбирает случайным образом ключ зашифрования в некоторой криптосистеме с открытым ключом. В таком случае совершенно неважно, что нахождение соответствующего ключа расшифрования будет трудновычислимой задачей в некоторых достаточно редких случаях, а в большинстве случаев — легкорешаемой.

В криптографии часто используются *вероятностные* или *стохастические* алгоритмы. Интуитивно это означает, что в процессе выполнения алгоритма на некоторых этапах выполняется случайный выбор (т. е. может вызываться некоторый генератор случайных чисел). Введенная выше терминология расширяется и на стохастический случай. Так, мы можем говорить об алгоритмах, работающих в *случайном полиномиальном времени*. Соответствующий класс проблем часто обозначается как BPP. Общепринятым считается, что $BPP \neq NP$. Стохастические алгоритмы могут и не привести к нужному результату, однако вероятность неудачи может быть сделана произвольно малой. Обычно временная сложность возрастает, когда вероятность неудачи становится меньше. Сама неудача здесь есть следствие стохастики. Следующая терминология используется, чтобы указать на разные типы неудач. *Алгоритм Монте-Карло* может привести к неверному результату в каких-то случаях. *Алгоритм Лас-Вегас* всегда дает правильный результат, но в некоторых случаях он может остановиться с ответом “Я не знаю”.

В заключение упомянем, что, говоря о временной сложности, мы обычно рассматриваем не вычислительные шаги машины Тьюринга, а некоторые другие элементарные операции, такие, как битовое умножение. Классы P и NP инвариантны относительно таких изменений, хотя, например, степени и коэффициенты возникающих многочленов могут изменяться.

Приложение Б

Элементы теории чисел

Это приложение состоит из обзора теоретико-числовых результатов, используемых в этой книге. Большинство доказательств очень просты и могут быть найдены, например, в [Ко].

Целое число a *делит* другое целое число b, символически $a|b$, если и только если $b = da$ для некоторого целого числа d. В этом случае a называется *делителем* или *множителем* b. Пусть a — целое число, большее 1. Тогда a — *простое* число, если его единственными положительными делителями будут 1 и само a, в противном случае a называется *составным*. Любое целое $n > 1$ может быть представлено единственным образом с точностью до порядка сомножителей как произведение простых. Существенный с точки зрения криптографии факт состоит в том, что не известно никакого эффективного алгоритма *разложения* чисел на множители, хотя, с другой стороны, не было получено и никакой нетривиальной нижней оценки временной сложности разложения. Никаких эффективных методов не известно даже в таком простом случае, когда необходимо восстановить два простых числа p и q из их произведения $n = pq$.

Наибольший общий делитель a и b, обозначение — НОД(a, b) или просто (a, b), есть наибольшее целое, делящее одновременно и a и b. В эквивалентной форме (a, b) есть то единственное натуральное число, которое делит a и b и делится на любое целое, делящее и a и b. Аналогично, *наименьшее общее кратное*, НОК (a, b), есть наименьшее натуральное число, делящееся и на a и на b.

Наибольший общий делитель может быть вычислен с помощью ал-

горитма Евклида. Он состоит из следующей цепочки равенств:

$$\begin{array}{lcll} a & = & bq_1 + r_1 \ , & 0 < r_1 < b \ , \\ b & = & r_1 q_2 + r_2 \ , & 0 < r_2 < r_1 \ , \\ r_1 & = & r_2 q_3 + r_3 \ , & 0 < r_3 < r_2 \ , \\ & \vdots & & \\ r_{k-2} & = & r_{k-1} 1 q_k + r_k, & 0 < r_k < r_{k-1}, \\ r_{k-1} & = & r_k q_{k+1}. & \end{array}$$

Остановка гарантируется, поскольку остатки от деления r_i образуют строго убывающую последовательность натуральных чисел. Из этой цепочки немедленно получаем, что r_k есть общий делитель a и b и более того, что любой общий делитель a и b делит, в свою очередь, r_k. Таким образом, $r_k = (a, b)$.

Оценим теперь временную сложность этого алгоритма. Легко видеть, что алгоритм, выполняющий обычное деление, работает за квадратичное время. Итоговая оценка была бы все еще экспоненциальной, если бы выполнялось только $r_{i+1} < r_i$. К счастью, легко видеть, что $r_{i+2} < r_i/2$ для всех i. Это дает верхнюю оценку $2 \log_2 a$ для числа равенств. Таким образом, временная сложность в целом самое большее кубическая.

Считывая цепочку равенств снизу вверх, найдем суммарно за кубическое время целые числа x и y, такие, что

$$(a, b) = xa + yb \ .$$

Два целых a и b *взаимно просты*, если $(a, b) = 1$. *Функция Эйлера* $\varphi(n)$, $n \geq 1$, определяется как число неотрицательных $a < n$, таких, что a и n взаимно просты. Имеем $\varphi(1) = 1$ и $\varphi(p^b) = p^b - p^{b-1}$, где p — простое и $b \geq 1$. Легко также видеть, что $\varphi(mn) = \varphi(m)\varphi(n)$, если m и n взаимно просты. Опираясь на эти факты, можно вычислять $\varphi(n)$ для любого n. Это вычисление будет эффективным, если знать разложение n.

Говорим, что a *сравнимо* с b *по модулю* m,

$$a \equiv b \pmod{m} \ ,$$

если m делит разность $a - b$. Число m называется *модулем*. Предполагается, что $m \geq 2$. Для любого целого x, в точности одно из чисел $0, 1, \ldots, m-1$ сравнимо с x по модулю m. Так определенное число называется *наименьшим неотрицательным остатком* x по модулю m и обозначается

$$(x, \bmod m) \ .$$

Это обозначение часто появляется в этой книге в различных контекстах. Обозначим далее через $[x]$ целую часть x, т. е. наибольшее целое $\leq x$. Имеем,

$$(x, \bmod m) = x - \left[\frac{x}{m}\right] \cdot m \ .$$

Мы уже видим, что если a и m взаимно просты, то тогда существуют x и y, такие, что $1 = xa + ym$. Отсюда $xa \equiv 1 \pmod m$. Целое число x будем называть *обратным* к a по модулю m и обозначать $a^{-1} \pmod m$. Обратное число определяется однозначно, если считать сравнимые числа равными. Сложность нахождения обратного числа примерно такая же, как и у алгоритма Евклида. Отсюда следует, что также и сравнение

$$az \equiv b \pmod m, \quad (a, m) = 1$$

может быть решено за кубическое время. Для нахождения z сперва вычисляем $a^{-1} \pmod m$ и умножаем его на b.

Если $(a, m) = 1$, то согласно *теореме Эйлера*

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

Если m простое число, не делящее a, этот результат принимает вид

$$a^{m-1} \equiv 1 \pmod m$$

и называется *малой теоремой Ферма*.

Если модули m_i попарно взаимно просты, то система сравнений

$$x \equiv a_i \pmod{m_i}, \quad i = 1, \ldots, k \ ,$$

имеет решение x, единственное с точностью до сравнений по модулю $M = m_1 \ldots m_k$. Этот результат, известный как *китайская теорема об остатках*, устанавливается в параграфе 6.3.

Поле F есть множество, на котором определены операции сложения и умножения, удовлетворяющие обычным требованиям: ассоциативности, коммутативности, дистрибутивности, существования аудитивного 0 и мультипликативной 1, аддитивных обратных и мультипликативных обратных для всех элементов за исключением 0. Рациональные числа и действительные числа образуют поля.

Конечные поля $F(q)$ с q элементами играют важную роль в криптографии. Легко видеть, что $q = p^h$, для некоторого простого p и $h \geq 1$. Удобный способ представления элементов поля $F(q)$ приводится в параграфе 3.5.

Обозначим через $F^*(q)$ множество всех ненулевых элементов $F(q)$. Некоторый элемент g из $F^*(q)$ называется *образующей* или *порождающим элементом* $F^*(q)$, если для всех a из $F^*(q)$ найдется такое целое x, что $g^x = a$ в поле $F^*(q)$. Всего имеется $\varphi(q-1)$ образующих g. Число x при этом будет *дискретным логарифмом* a по основанию g. Известно, что вычисление дискретных логарифмов (когда g, a и q заданы) примерно такая же труднорешаемая задача, как и разложение на множители.

Рассмотрим некоторое простое $p > 2$. Если элемент a из $F^*(q)$ есть квадрат, т. е. $a = x^2$ для подходящего x, то a называется *квадратичным вычетом* по модулю p. В противном случае a называется *квадратичным невычетом* по модулю p. Понятно, что a, $1 \le a \le p-1$, будет квадратичным вычетом по модулю p тогда и только тогда, когда сравнение

$$x^2 \equiv a \pmod{p}$$

имеет решение. В таком случае также и $-x$ будет решением, т. е. a имеет два *квадратных корня* по модулю p. Все квадратичные вычеты находятся возведением в квадрат элементов 1, 2, ..., $(p-1)/2$. Таким образом, имеется всего по $(p-1)/2$ квадратичных вычетов и квадратичных невычетов.

Символ Лежандра в случае целого a и простого $p > 2$ определяется соотношением

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{если } p \text{ делит } a, \\ 1, & \text{если } a \text{ — квадратичный вычет по модулю } p, \\ -1, & \text{если } a \text{ — квадратичный невычет по модулю } p\ . \end{cases}$$

Понятно, что a можно заменить любым целым числом, сравнимым с a (mod p), не изменяя значения символа Лежандра. Элементарный результат о символе Лежандра есть

$$(*) \qquad \left(\frac{a}{p}\right) = a^{(p-1)/2} \pmod{p}\ .$$

Символ Якоби является обобщением символа Лежандра. Рассмотрим целое a и нечетное $n > 2$. Далее, пусть $n = p_1^{i_1} \dots p_k^{i_k}$ есть разложение n на простые множители. Тогда символ Якоби определяется как произведение соответствующих символов Лежандра:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{i_1} \dots \left(\frac{a}{p_k}\right)^{i_k}\ .$$

Ясно, что и здесь a может быть заменено без изменения символа Якоби на число, сравнимое с $a \pmod{n}$. Из (*) легко вытекает свойство мультипликативности

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right) \cdot \left(\frac{b}{n}\right).$$

Следовательно,

$$\left(\frac{ab^2}{n}\right) = \left(\frac{a}{n}\right).$$

Для некоторых значений a символ Якоби вычисляется так:

$$\left(\frac{1}{n}\right) = 1, \ \left(\frac{1}{n}\right) = (-1)^{(n-1)/2}, \ \left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}.$$

При вычислении символа Якоби основное сведение выполняется на основе закона взаимности:

$$\left(\frac{m}{n}\right) = (-1)^{(m-1)(n-1)/4} \left(\frac{n}{m}\right),$$

где m и n — нечетные числа больше 2. В эквивалентном виде

$$\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right),$$

если только не выполняется

$$m \equiv n \equiv 3 \pmod{4},$$

а в этом случае

$$\left(\frac{m}{n}\right) = -\left(\frac{n}{m}\right).$$

Значение $\left(\frac{m}{n}\right)$ может быть теперь найдено без разложения на множители следующим образом (за исключением случая степеней 2). При необходимости m заменяется на $(m, \mathrm{mod} n)$; аналогичная замена осуществляется также и на последующих шагах процедуры. Применение закона взаимности позволяет уменьшить "знаменатель" в $\left(\frac{m}{n}\right)$. Как и в случае алгоритма Евклида, это уменьшение на одном шаге может быть малым, однако два последовательных шага сокращают знаменатель по крайней мере в два раза. В итоге это дает примерно ту же оценку сложности вычисления $\left(\frac{m}{n}\right)$, как и в случае алгоритма Евклида. Пример вычисления приводится в параграфе 6.5.

Если p — простое, то описанный метод является быстрым алгоритмом также и для определения: будет ли данное число a квадратичным вычетом или невычетом по модулю p. Никаких подобных быстрых алгоритмов неизвестно в случае, когда вместо простого p имеют дело с

произвольным n. Рассмотрим более детально важный для криптографии случай, когда n есть произведение двух простых чисел, $n = pq$.

Как отмечалось выше, половина из чисел $1, \ldots, p-1$ является квадратичными вычетами по модулю p, а другая половина — невычетами. Конечно, аналогичное утверждение верно и для q. С другой стороны, некоторое число a будет квадратичным вычетом по модулю n, т. е. $x^2 \equiv a \pmod{n}$ для подходящего x, если и только если a будет квадратичным вычетом одновременно и по модулю p, и по модулю q. Все это означает, что в точности половина из чисел a

$$0 < a < n \text{ и } (a, n) = 1$$

удовлетворяет равенству $\left(\frac{a}{n}\right) = +1$, а для другой половины выполняется $\left(\frac{a}{n}\right) = -1$. Более того, половина из чисел a, удовлетворяющих равенству $\left(\frac{a}{n}\right) = +1$, будут квадратичными вычетами по модулю n, а именно те, для которых

$$\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = +1 \ .$$

Другая половина, а именно те, для которых

$$\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$$

будут невычетами. И похоже, что нет способа выяснить, какой из этих двух случаев имеет место, если только n не будет разложено на множители.

Предположим, нам известно, что a, $0 < a < n$, является квадратичным вычетом по модулю n. Тогда для некоторого x

$$x^2 \equiv a \pmod{n} \ .$$

Нахождение x, т. е. извлечение квадратных корней по модулю n является весьма важной задачей в криптографии. Давайте снова рассмотрим случай $n = pq$. По предположению, a является квадратичным вычетом как по модулю p, так и по модулю q. Из этого вытекает существование чисел y и z, таких, что

$$(\pm y)^2 \equiv a \pmod{p} \text{ и } (\pm z)^2 \equiv a \pmod{q} \ .$$

Более того, y и z могут быть найдены за полиномиальное время (со степенью полинома не выше 4), при условии, что p и q известны. Детально такой алгоритм описан, например, в [Ko]. В алгоритме предполагается

известным некоторый невычет по модулю p, а также некоторый невычет по модулю q. Такие невычеты могут быть быстро найдены с помощью стохастического алгоритма.

Из сравнений

$$x \equiv \pm y \pmod{p} \text{ и } x \equiv \pm z \pmod{q}$$

по китайской теореме об остатках теперь можно получить четыре квадратных корня x по модулю n. Эти квадратные корни можно записать в виде $\pm u$ и $\pm w$, где $u \not\equiv \pm w \pmod{n}$. Назовем такие u и w *различными* квадратными корнями. Следующие два факта важны для криптографии. Знание двух различных квадратных корней позволяет разложить n. Действительно,

$$u^2 - w^2 = (u+w)(u-w) \equiv 0 \pmod{n} .$$

Это означает, что n делит $(u+w)(u-w)$. Однако по выбору u и w n не делит ни $u+w$, ни $u-w$. Отсюда следует, что наибольший общий делитель $u+w$ и n (быстровычисляемый алгоритмом Евклида) есть p или q.

Второй важный факт состоит в том, что при $p \equiv q \equiv 3 \pmod{4}$ два различных квадратных корня u и w из одного и того же числа a по модулю n имеют различные символы Якоби:

$$\left(\frac{u}{n}\right) = -\left(\frac{w}{n}\right) .$$

Это следует из того, что, как было показано выше, или

$$u \equiv w \pmod{p} \text{ и } u \equiv -w \pmod{q} ,$$

или

$$u \equiv -w \pmod{p} \text{ и } u \equiv w \pmod{q} ,$$

а по предположению относительно p и q

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

Задачи

1. Зашифруйте исходный текст

   DONOTGOTOSAUNASOONAFTEREATING

   используя криптосистему Цезаря с ключевым словом SUPERDOG и числом 9.

2. Исходный текст SAUNA зашифрован как TAKE BACK VAT OR BONDS. Опишите используемую криптосистему.

3. Исходный текст SAUNAANDLIFE зашифрован как RMEMHCZZ-TCEZTZKKDA. Опишите используемую криптосистему.

4. В криптосистеме Хилла (см. пример 1.2) с матрицей

   $$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}.$$

   зашифруйте текст PAYMOREMONEY.

5. Матрица теперь

   $$\begin{pmatrix} 11 & 2 & 19 \\ 5 & 23 & 25 \\ 20 & 7 & 1 \end{pmatrix}.$$

   Зашифруйте STOPPAYMENTX.

6. Сформулируйте необходимое и достаточное условие обратимости матрицы M, когда арифметические операции выполняются по модулю 26. (Это требуется в криптосистеме Хилла.) Найдите обратную матрицу для нескольких матриц порядка 2.

7. Известно, что при зашифровании использовалась криптосистема Хилла с матрицей 2-го порядка. Наиболее часто встречающиеся в криптотексте диграммы — RH и NI, в то время как в исходном языке они TH и HE. Какая матрица может быть вычислена из этой информации?

8. Для зашифрования вначале используется матрица $\begin{pmatrix} 2 & 3 \\ 1 & 17 \end{pmatrix}$, а затем к результату применяется матрица $\begin{pmatrix} 5 & 1 \\ 25 & 4 \end{pmatrix}$. Постройте одну матрицу с тем же действием.

9. Условие, как и в задаче 8, но теперь матрицы (в указанном порядке)
$$\begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

10. В более общем случае, если исходные матрицы имеют порядки m и n, то каков будет порядок матрицы с объединенным действием?

11. Криптосистема *замкнута относительно композиции*, если для любых двух ключей зашифрования найдется один ключ, обладающий эффектом последовательного применения этих двух ключей. Замкнутость относительно композиции означает, что последовательное применение двух ключей не добавляет безопасности. Предыдущие задачи показывают, что криптосистема Хилла замкнута относительно композиции. Изучите это свойство применительно к некоторым другим криптосистемам, рассматриваемым в книге.

12. В простых криптосистемах каждый ключ зашифрования может быть представлен как композиция нескольких *порождающих ключей*. В криптосистеме Цезаря такой порождающий ключ — это E_1 — ключ, отображающий каждую букву в следующую. В аффинной системе буква x, $0 \le x \le 25$, отображается в букву $(ax + b, \mathrm{mod} 26)$, где $(a, 26) = 1$. Покажите, что никакой ключ не может быть порождающим для аффинной системы, в то время как два ключа могут.

13. Расшифруйте следующий криптотекст, предложенный участникам конференции EUROCRYPT-88 в Давосе:

EXVITL AMSYMX EAKSSI KIRZMS
YEKDAY OSINAL PVITHE RRJMLO
OIEUSM GPLKSM ADAVOS LULRVK
SIXMTA IDAVOS

14. Название какого города из четырех букв зашифровано как BHFLYPBT, если был использован следующий алгоритм зашифрования. Вначале произвольная "мусорная" буква добавлялась после каждой буквы исходного текста. (Таким образом, в результирующем слове 2-я, 4-я, 6-я и 8-я буквы не существенны.) Затем использовалась криптосистема Хилла с матрицей 2-го порядка, в которой слово AIDS шифруется как слово AIDS.

15. Исходный алфавит {A, B, C, D}. Используется моноалфавитная система, в которой индивидуальные буквы шифруются так:

$$\mathrm{A} \to \mathrm{BB}\ , \quad \mathrm{B} \to \mathrm{AAB}\ , \quad \mathrm{C} \to \mathrm{BAB}\ , \quad \mathrm{D} \to \mathrm{A}\ .$$

Например, слово ABDA шифруется как BBAABABB. Покажите, что расшифрование всегда однозначно. Покажите, что оно не будет однозначным, если буквы зашифровать так:

$$\mathrm{A} \to \mathrm{AB}\ , \quad \mathrm{B} \to \mathrm{BA}\ , \quad \mathrm{C} \to \mathrm{A}\ , \quad \mathrm{D} \to \mathrm{C}\ .$$

16. Дополнение $\neg x$ бита x определяется обычным образом:$\neg 0 = 1$ и $\neg 1 = 0$. Докажите, что в криптосистеме DES замена в исходном тексте и в ключе каждого бита на его дополнение приводит к замене каждого бита на его дополнение в криптотексте.

17. Любое слово над алфавитом {A, B} может возникнуть как исходный текст. Первый моноалфавитный ключ зашифрования определяется соотношениями

$$\mathrm{A} \to \mathrm{CCD}\ , \quad \mathrm{B} \to \mathrm{C}\ ,$$

а второй ключ — соотношениями

$$\mathrm{A} \to \mathrm{C}, \quad \mathrm{B} \to \mathrm{DCC}\ .$$

Какие слова над {A, B} шифруются как одно и то же слово над {C, D} для обоих ключей?

18. Наиболее часто встречающиеся триграммы в некотором криптотексте есть LME, WRI и ZYC, в то время как в исходном языке они THE, AND и THA. Какая матрица использовалась в криптосистеме Хилла?

19. Каждая буква x, $0 \le x \le 25$, шифруется как $(f(x), \mathrm{mod}26)$, где $f(x)$ — квадратный трехчлен. Найдите его, если известно, что три наиболее часто встречающиеся буквы в криптотексте Z, V, B (в этом порядке), в то время как в исходном языке это E, T, N.

20. Рассмотрим один очень слабый вариант криптосистемы с одноразовым блокнотом, описанным в конце параграфа 1.3. В качестве основной книги будет использоваться эта книга. Например, ключ 12345 означает пятую букву четвертого слова из третьего абзаца из параграфа 1.2. Зашифруйте текст RACCOONDOGANDSAUNA, используя ключ 43333.

21. И ключевое слово, и исходный текст могут считываться из таблиц Виженера и Бьюфорта различными способами. Запишите арифметические выражения для некоторых возникающих отображений.

22. Простую криптосистему, основанную на подстановках, можно задать так. Фиксированная подстановка на множестве чисел $\{1,\ 2, \ldots,\ n\}$ применяется к каждому блоку. Например, SAUNA превращается в UNSAA, если $n = 5$ и подстановка меняет местами первую и третью буквы, а также вторую и четвертую, оставляя пятую букву на месте. Покажите, что такой же результат всегда можно получить, используя подходящую криптосистему Хилла.

23. Всякая криптосистема порождает отображение из множества слов исходных текстов в множество слов криптотекстов. В целом о таких отображениях известно бывает очень много, но, например, отображение, индуцированное системой Цезаря, охарактеризовать легко. Рассмотрите различные криптосистемы и выясните, будут ли индуцированные отображения сохранять длины слов?

24. Дайте необходимые и/или достаточные условия реализуемости отображения с помощью квадрата Плейфейра. Этот результат позволит сконструировать “осмысленные переводы” типа того, что приведен в книге.

25. Объясните разницу (кроме разницы в размерах алфавитов) между отображениями, реализуемыми с помощью квадрата Плейфейра и прямоугольника Плейфейра размера 3×9.

26. Условие то же, что и в задаче 24, но теперь для колеса Джефферсона. Обратите особое внимание на важную роль расстояния между строчками исходного текста и криптотекста.

27. Какой период получается у кулачковой матрицы и ступенчатой матрицы из текста книги?

28. Постройте кулачковую и ступенчатую матрицы, приводящие к периоду 17 (соответственно $19 \cdot 21$).

29. Постройте кулачковую и ступенчатую матрицы, приводящие к максимальному периоду (можно посмотреть в [BeP]).

30. Покажите, что вектор A' размерности 10 из параграфа 2.1 является инъективным, т.е. не найдется такого α, что задача о рюкзаке (A', α) имела бы два решения.

31. Пусть $A = (a_1, \ldots, a_n)$ — рюкзачный вектор, т.е. a_i являются различными натуральными числами. Некоторое натуральное число α *представляется* при помощи A, если α можно выразить суммой a_i, причем никакое a_i не появляется в сумме дважды. Если A — инъективный вектор, то тогда ясно, что $2^n - 1$ чисел представляются при помощи A. Это наибольшее из возможных значений. Каким будет наименьшее из возможных значений в терминах n?

32. Для заданной задачи о рюкзаке $(A,\ k)$ требуется найти все решения. Покажите, что эта проблема даже не из NP.

33. Почему число 2047 будет неудачным выбором для модуля в криптосистеме RSA, кроме того, конечно, что это число слишком мало?

34. Покажите, что экспоненты зашифрования и расшифрования будут совпадать, если модуль в RSA равен 35.

35. Некоторые блоки исходного текста остаются неизмененными при зашифровании в RSA. Покажите, что их число будет

$$\big(1 + (e-1, p-1)\big)\big(1 + (e-1)(q-1)\big)\ .$$

36. Постройте пример применения алгоритма Шамира, когда для u/m отыскиваются по крайней мере два различных интервала. Можно ли что-нибудь сказать в общем случае о числе различных интервалов. Возможно ли, что интервал вырождается в точку?

37. Докажите, что вектор $(i,\ i-1,\ i-2, \ldots,\ i-j)$, $i - j \geq 1$ будет супердостижимым только в случае, если $j = 2$ и $i \geq 4$.

38. Вектор (7, 3, 2) является ((7, 15, 38), 73, 84) супердостижимым. Примените технику леммы 3.5 для получения достаточно маленького сомножителя.

39. Докажите, что любой инъективный вектор $(b_1,\ b_2,\ b_3)$ будет перестановочно-супердостижимым.

40. Опишите алгоритм нахождения наименьшего модуля m, такого, что данный сверхрастущий вектор будет $(A,\ t,\ m)$-супердостижимым.

41. Рассмотрите все рюкзачные векторы с компонентами ≤ 4. Покажите, что являются супердостижимыми следующие векторы:

$$(2,\ 4,\ 3),\quad (4,\ 3,\ 2),\quad (1,\ 2,\ 4),\quad (2,\ 4,\ 1),\quad (4,\ 1,\ 2)\ .$$

42. Докажите, что векторы (5, 3, 4) и (5, 4, 3) являются единственными супердостижимыми векторами среди векторов с компонентами 3, 4, 5.

43. Выразите элементы поля $F(27)$ через корень многочлена, неприводимого над $F(3)$. Найдите порождающий элемент и постройте таблицу дискретных логарифмов.

44. Проведите криптоанализ криптосистемы, основанной на плотных рюкзаках, в случае когда известна некоторая информация о секретной лазейке. (Здесь следует обратиться к [Cho].)

45. Рассмотрите первую часть ($n = 55$) примера 4.1. Пошлите подписанное сообщение пользователю, чья открытая экспонента зашифрования есть 13. (Имеем $e = 7$, $d = 23$.)

46. Покажите, что число 3215031751 — составное и сильно псевдопростое по каждому из оснований 2, 3, 5, 7.

47. Рассмотрите общий метод обмена ключами, указанный в самом конце гл. 4, в случае некоторых конкретных функций f. Можно ли улучшить отношение m/m^2 между работой, выполняемой легальным пользователем, и работой, выполняемой криптоаналитиком.

48. Предположим, что имеются алгоритмы для SQUAREFREENESS (см. параграф 2.2) и нахождения $\varphi(n)$. Можно ли свести один из них к другому?

49. В функциональной криптосистеме 3 есть начальное значение, а функции $f_0(x) = 3x$ и $f_1(x) = 3x + 1$. Так, 011 зашифруется как

$$3f_0f_1f_1 = 85\ .$$

Какой имеется простой способ расшифровать криптотекст, записанный как десятичное число? Какие числа могут появляться в качестве криптотекстов?

50. Покажите, что рюкзачный вектор

$$(2106,\ 880,\ 1320,\ 974,\ 2388,\ 1617,\ 1568,\ 2523,\ 48,\ 897)$$

будет супердостижимым.

51. Приведите пример задачи о рюкзаке $(A(i), \alpha(i))$, имеющей ровно i решений, $i = 1, 2, \ldots$.

52. Аналогично примеру 3.5, пусть открытой информацией будут $A = (1,\ 2,\ 3,\ 0,\ 0,\ 4)$ (A рассматривается как вектор-столбец) и $m = 7$. Секретная матрица есть

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}\ .$$

Что будет подписью для исходного текста 3, (1) если использовать основной способ, (2) если использовать случайный вектор $(1,\ 0,\ 0,\ 0,\ 1,\ 1)$?

53. Ясно, что может быть построена двойственная теория для убывающих и сверхубывающих векторов, которые определяются аналогично возрастающим и сверхвозрастающим векторам. В частности, понятие супер-d-достижимости относится к сверхубывающим векторам. Приведите примеры инъективных векторов, которые не будут ни супердостижимыми, ни супер-d-достижимыми.

54. Постройте протокол для бросания кубика по телефону. Не стоит удовлетворяться следующим очевидным решением. Подбрасываем монету три раза. В случае орел-орел-орел или решка-решка-решка перебрасываем до получения любого другого результата.

55. Предположим, что простые числа p и q в RSA имеют по 100 цифр, причем первая цифра у них не равна нулю. Оцените число возможных значений для n.

56. YJCVKUVJGJGCTVQHUCWPC? UVQXG.

57. Докажите, что остатки в алгоритме Евклида удовлетворяют неравенствам $r_{j+2} < r_j/2$ для всех j. Постройте вариант этого алгоритма, в котором допускались бы отрицательные остатки и сходимость была бы немного лучше: $r_{j+2} \leq r_{j+1}/2$.

58. Расшифруйте

    KOKOOKOKOONKOKOKOKKOKOKOKOKKOKOKOKOKOKKO

    и

    Оба выражения на самом деле являются предложениями в известных естественных языках. Несомненно, что язык исходных текстов играет какую-то роль!

59. Рассмотрите исходный текст длины 47, приведенный в примере шифрования на машине С-36. Если добавить к концу исходного текста YES, то как будет продолжен криптотекст?

60. Пусть $(a, m) = 1$. Покажите, что $a^{\varphi(m)/2} \equiv 1 \pmod{m}$ при условии, что m отлично от чисел 1, 2, 4, p^k и $2p^k$, где p — простое число и $k \geq 1$.

61. Докажите, что $(a^m - 1, a^n - 1) = a^{(m,n)} - 1$. Предполагается, что $a > 1$.

62. В системе RSA всегда найдутся такие экспоненты зашифрования, что любой текст не меняется при зашифровании. Более точно, докажите следующее утверждение. Для любых p и q экспонента e может быть выбрана так, что $w^e \equiv w \pmod{n}$ для всех w. (Тривиальные случаи $e = 1$ и $e = \varphi(n) + 1$ исключаются.)

63. Следующий алгоритм зашифрования является классическим и он был проиллюстрирован на рис. 2.4. Большое простое число p известно всем пользователям. Каждый пользователь выбирает и держит в секрете экспоненты зашифрования и расшифрования, e и d, причем $ed \equiv 1 \pmod{p-1}$. Таким образом, A шифрует исходный текст w как

$$E_A(w) = (w^{e_A}, \mathrm{mod} p) \ .$$

Сначала A посылает B криптотекст $E_A(w) = c$. B отвечает A сообщением $E_B(c) = c_1$. Наконец, A посылает $D_A(c_1)$ к B. Покажите, что B в состоянии осуществить расшифрование, и рассмотрите возникающие здесь вопросы безопасности.

64. Найдите необходимые и достаточные условия на p и t, чтобы любой элемент $\neq 0, 1$ поля $F(p^t)$ был бы (a) порождающим элементом, (b) квадратом некоторого порождающего элемента.

65. Предположим, что экспонента зашифрования e в системе RSA является небольшой. Предположим, что некий оракул всегда сообщает $E(x + r)$ для заданных $E(x)$ и r. (Понятно, что никакого оракула не нужно, чтобы сообщить $E(x \cdot r)$ по заданным $E(x)$ и r.) Как в таком случае можно осуществить расшифрование?

66. Разложить $n = 4386607$, зная $\varphi(n) = 4382136$.

67. Рассмотрим следующую модификацию RSA, когда модуль n есть произведение трех больших чисел p, q и r. В этом случае попрежнему выполняется $ed \equiv 1 \pmod{\varphi(n)}$. Отметьте преимущества и недостатки этой модификации в сравнении с обычной системой RSA.

68. Пусть $f(x)$ и $g(x)$ есть односторонние функции. Приведите эвристические аргументы в пользу того, что ни одна из функций $f(x)+g(x)$, $f(x)\cdot g(x)$ не обязательно должна быть односторонней.

69. Покажите, что следующая проблема лежит в пересечении классов NP и $Co\text{-}NP$ для любой криптосистемы. По заданному криптотексту требуется выяснить, будет или нет SUVI подсловом в соответствующем исходном тексте.

70. Рассмотрите последний случай из примера 4.2, где $n = 8137$. Вычислите таблицу для $r(i)$, $ANS(i)$ и $t(i)$, при $x = 20$. Прочитайте замечания в конце этого примера.

71. В системе DES каждая S-матрица переводит 6-битовый вход в 4-битовый выход. Покажите, что изменение одного входного бита всегда приводит к изменению по крайней мере двух выходных битов.

72. Если зафиксировать два входных бита, то каждая S-матрица определяет отображение 4-битовых наборов в 4-битовые наборы. Какой бит можно зафиксировать, чтобы получить взаимнооднозначное отображение во всех восьми случаях? Приведите

пример отображения такого типа, не являющегося взаимно-однозначным.

73. Докажите, что имеется бесконечно много пар простых чисел (p, q), таких, что $p \equiv q \equiv 3 \pmod 4$, но $p \not\equiv q \pmod 8$. Используйте теорему Дирихле, утверждающую, что в последовательности $ia + b$, $i = 1, 2, \ldots$, найдется бесконечно много простых чисел, при условии, что a и b — натуральные числа и $(a, b) = 1$.

74. Рассмотрим вектор рюкзака $A = (a_1, \ldots, a_n)$, где $a_i = P/p_i$, $i = 1, \ldots, n$ и p_i — различные простые числа, произведение которых равно P. Укажите простой алгоритм для решения задачи о рюкзаке (A, α).

75. Постройте криптосистему Уильямса для $p = 47$, $q = 59$, часто упоминавшимся в тесте. Зашифруйте 1991. См. параграф 5.1.

76. Предположим, что в RSA $p = 127$ и $q = 131$. Сколько сообщений шифруются в себя для обеих экспонент 29 и 31?

77. Рассмотрите систему обмена ключей Диффи-Хеллмана (параграф 4.6) с $q = 4079$, $q = 1709$ и секретными числами $k_1 = 2344$ и $k_2 = 3420$. Какие числа раскрываются и каким будет общий ключ между пользователями A_1 и A_2?

78. Найдите все квадратные корни из 64 по модулю 105.

79. В схеме Эль Гамаля, рассмотренной в конце параграфа 4.6, открываются простое число q и порождающий элемент g поля $F^*(q)$. Что произойдет при зашифровании и расшифровании, если в действительности g не будет порождающим элементом?

80. В *шестнадцатеричном* представлении 4-битовых наборов 0000, 0001, ..., 1111 используются цифры 0, 1, 2,..., 9, A, B, C, D, E, F в указанном порядке. Предположим, что ключом в системе DES будет 0123456789ABCDEF. Зашифруйте тексты 5_{16} и 6_{16}.

81. Изучите криптографическое значение начальной перестановки в DES.

82. Укажите все квадратичные вычеты по модулю 29, а также по модулю 31. Докажите, что для всех простых p число 3 будет квадратичным вычетом по модулю p тогда и только тогда, когда $p \equiv 1 \pmod 3$.

83. Пусть n будет таким, как в RSA. Покажите, что проблема перечисления всех квадратичных вычетов по модулю n даже не из NP.

84. Рассмотрим схему идентификации из примера 6.5, но теперь $n = 2491$. Секретный идентификатор P состоит из тройки

$$c_1 = 143, \quad c_2 = 32, \quad c_3 = 2261 \ .$$

Опишите один этап протокола для выбранных значений $r = 61$ и $S = \{1, 3\}$.

85. Докажите лемму 5.1.

86. Рассмотрим основанную на итерации морфизмов систему с начальными морфизмами

$$\begin{array}{lllll} h_0: & a \to ac & b \to ba & c \to ca \ , \\ h_1: & a \to aa & b \to bc & c \to cb \ , \end{array}$$

и с начальным словом c. Покажите, что легальный получатель может расшифровать следующим образом. Вначале к криптотексту применяется интерпретирующий морфизм и получается слово w над алфавитом $\{a, b, c\}$. Строится слово u, i-я буква которого есть $(2^{i-1} + 1)$-я буква w. Слово u читается справа налево с заменой a и b на 0 и 1 соответственно. (Слово u не содержит c.)

87. Покажите, что нахождение секретной лазейки для криптосистем, основанных на итерации морфизмов, есть NP-полная проблема. См. [Kar3].

88. Приведите причины, по которым декодирование для кодов Гоппы существенно проще, чем для линейных кодов.

89. Рассмотрим протокол для игры в покер по телефону, обсуждаемый в конце параграфа 6.2. Какие имеются возможности схитрить, если некоторые из выбираемых чисел p_i и q_i на самом деле не будут простыми? (См. обсуждение бросания монеты по телефону.)

90. Придумайте метод разделения секретов, основанный на некоторых других идеях, а не на китайской теореме об остатках.

91. Придумайте протокол голосования (как в параграфе 6.4) для случая двух сверхдержав и пяти обычных стран.

92. A располагает 8 секретами и желает раскрыть B в точности один из них таким образом, чтобы только B знал, какой из секретов был ему передан. Однако B не может решить, какой секрет он хочет получить. Придумайте протокол для этой ситуации.

93. Приведите конкретный числовой пример для 7-шагового протокола, описанного после примера 6.3. Рассмотрите возможности активного мошенничества в этом протоколе.

94. Опишите подробно протокол, рассмотренный в параграфе 6.6 для тринадцати избирателей и двух стратегий голосования: (a) с двумя агентствами C и L, (b) только с одним агентством C.

95. Придумайте протокол для доказательства с нулевым знанием, что вам известно решение в данной задаче о рюкзаке. Желательно использовать идею запирающихся ящиков.

96. Рассмотрим задачу коммивояжера в следующем виде, когда для данной карты с указанием всех расстояний и данного k требуется найти маршрут через все города на карте длиной $\leq k$. Предложите протокол с нулевым знанием для убеждения проверяющей, что вам известно решение.

97. Рассмотрим некоторую аксиоматическую систему для пропозиционального исчисления и некоторую простую теорему, доказательство которой состоит, скажем, из пяти шагов. Предложите доказательство с нулевым знанием этой теоремы. Рассмотрите, будут или нет эти идеи распространяться на любые доказательства в произвольных формальных системах.

98. Используя RSA, получите метод построения запирающихся ящиков.

99. Рассмотрим РФПИ $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge \neg x_3$. Объясните, почему вы потерпите неудачу, если попытаетесь доказать с нулевым знанием, что вы знаете выполняющее приписывание переменных.

100. Приведите численный пример для второго протокола из параграфа 6.9 для его полной формы, когда A есть матрица. Обратитесь к [Sh5] для обобщений протокола и изучите возникающие здесь вопросы безопасности.

Исторические и библиографические сведения

Поскольку некоторым идеям в криптографии уже несколько тысяч лет, то не имеет особого смысла пытаться проследить первоисточники рассмотренных в гл. 1 фактов. [Ka] — это превосходный и полный справочник. В [Ga] рассматриваются криптоаналитические методы до эпохи компьютеров. Криптосистема из примера 1.2 была введена в [Hil]. [Kon] и [BeP] обсуждают различные криптоаналитические методы для классических систем. [Zim] может быть упомянута в качестве примера многочисленных книг по криптографии до эры открытых ключей.

Открытые ключи были введены в [DH]. Основная рюкзачная система, рассматриваемая в гл. 2, описана в [MeH], а возникающие там вопросы сложности — в [Br1] и [Kar1]. Покер по телефону, подбрасывание монеты по телефону и забывающая передача взяты из [ShRA], [Bl1] и [Rab 2] соответственно.

Теория, представленная в параграфах 3.2 и 3.3, сожержится в [Sh2], [Sa3] и [Sa4]. См. также [Adl]. Криптосистемы из параграфа 3.4 взяты (в этом порядке) из [EvY], [Sh3], [Sh1], а также введены Грэхемом и Шамиром. [Cho] является основным источником для плотных рюкзаков.

Теория, представленная в гл. 4, была инициирована в [RSA]. [Rab1] — более ранняя работа. См. в [Ko] оригинальные ссылки по параграфу 4.3. В параграфе 4.4 используются идеи из [Mil] и [Del]. Теорема 4.3 взята из [GMT]. См. также [SchA]. [Odl] — это довольно исчерпывающий труд о дискретных логарифмах, а [Ang] — хорошее резюме по сложности теоретико-числовых проблем.

Материал параграфа 5.1 взят из [Wil], а параграфа 5.2 — из [Sa2], [SaY], [Kar2] и [Kar3]. Криптосистемы, основанные на теории групп и регулярных языках, исходят из [WaM] и [Nie] соответственно. В [SiS]

также описана криптосистема, основанная на теории языков, а система, основанная на последовательностных машинах, восходит к [Ren]. Криптосистема из параграфа 5.4 была введена в [McE]. Схема подписи в конце параграфа 6.1 восходит к [Sh4], а материал параграфа 6.2 — к [Bl1] и [GM]. Метод разделения секрета, приведенный в параграфе 6.3, был предложен в [Mig]. Протокол для возраста из параграфа 6.4 взят из [Yao]. Понятие забывающей передачи восходит к [Rab2]. В параграфе 6.5 представлен простой протокол для секретной продажи секретов; более развитая техника содержится в [BCR]. Параграф 6.6 следует изложению [BuP] и [NaS]. Этой тематике посвящены многочисленные статьи, например, [Ben] — довольно полное исследование с несколько иными целями. [GMR] и [GMW] являются основополагающими статьями относительно доказательств с нулевым знанием. Первый протокол из параграфа 6.7 взят из [Dam]. Идеи из [Bl2] использовались в доказательстве теорем 6.2 и 6.3. Протокол для проблемы выполнимости, отличный от приводимого в теореме 6.5, дается в [BCC], где рассматриваются элементы соответствующих функциональных схем. В [DMP] и [BeG] рассматриваются неинтерактивные системы доказательств с нулевым знанием. Два доказательства, приведенные в параграфе 6.9, взяты из [FFS] (см. также [FiS]) и [Sh5]. Схемы мошенничества обсуждаются в [DGB].

Теоретико-информационная точка зрения, [Shan], в этой книге не обсуждается. Приводимый список литературы содержит только работы, на которые в этой книге делались ссылки. Дополнительные библиографические замечания содержатся, например, в [F1], [SP], [Br2], [Kra], [Til] и [Wel]. Cryptologia и Journal of Cryptology — журналы, посвященные криптографии. В других журналах также имеются статьи и целые номера о криптографии (например, майский 1988 г. выпуск журнала Proceedings of IEEE). CRYPTO и EUROCRYPT — ежегодные конференции, тезисы докладов которых обычно публикуются в Springer Lecture Notes in Computer Science. Кроме того, традиционные ежегодные конференции по теоретической кибернетике (STOC, FOCS, ICALP и т. д.) также содержат множество статей, посвященных криптографии.

Литература

[Adl] Adleman L. On breaking the iterated Merkle-Hellman public key cryptosystem. Proceedings of the 15th ACM Symposium on the Theory of Computing, 1983, p.p. 402–415.

[Ang] Angluin D. Lecture Notes on the Complexity of Some Problems in Number Theory.Yale University Computer Science Department Technical Report 243, 1982.

[BeP] Beker H. and Piper F. Cipher systems. Northwood Books, London, 1982.

[BeG] Bellare M. amd Goldwasser S. New paradigms for ditital signatures and message authentication based on non-interactive zero knowledge proofs. CRYPTO–89 Abstracts, Univesity of California, Santa Barbara, 1989, p.p. 189–204.

[Ben] Benaloh J.D.C. Verifiable secret-ballot elections. Yale University Computer Science Department Technical Report 561, 1987.

[Bl1] Blum M. Coin flipping by telephone. A protocol for solving impossible problems. SIGACT News, 1981, p.p. 23–27.

[Bl2] Blum M. How to prove a theorem so no one else can claim it. Proceedings of the International Congress of Mathematicians, 1987, p.p. 1444–1451.

[BC] Bose R.C. and Chowla S. Theorems in the additive theory of numbers. Comment.Math.Helvet. 37, 1962, p.p. 141–147.

[Br1] Brassard G. A note on the complexity og cryptography. IEEE Transactions on Information Theory IT-25, 1979, p.p. 232–233.

[Br2] Brassard G. Modern cryptology. Lecture Notes in Computer Science, vol.325, Springer, Berlin Heidelberg New York, 1988.

[BCC] Brassard G., Chaum D. and Crepeau C. An introduction to minimum disclosure. Amsterdam CWI Quarterly 1, 1988, p.p. 3–17.

[BCR] Brassard G., Crepeau C. and Robert J.-M. All-or-nothing disclosure of secrets. Lecture Notes in Computer Science, vol.236, Springer, Berlin Heidelberg New York, 1987, p.p. 234–238.

[BuP] Burk H. and Pfitzmann A. Digital payment systems enabling security and unobservability. Computers and Security 9, 1989, p.p. 399–416.

[Cho] Chor B.-Z. Two issues in public key cryptography. MIT Press, Cambridge, Mass., 1986.

[Cop] Coppersmith D. Fast evaluation of logarithms in fields of characteristic two. IEEE Transactions on Information Theory IT-30, 1984, p.p. 587–594.

[Dam] Damgaard I.B. On the existence of bit commitment schemes and zero-knowledge proofs. CRYPTO–89 Abstracts, University of California, Santa Barbara, 1989, p.p. 15–23.

[Del] Delaurentis J.M. A further weakness in the common modulus protocol for the RSA cryptoalgorithm. Cryptologia 8, 1984, p.p 253–259.

[De] Denning D.E. Cryptography and data security. Addison-Wesley, Reading, Mass., 1982.

[DMP] De Santis A., Micalo S. and Persiano G. Non-interactive zero-knowledge proof systems. Lecture Notes in Computer Science, vol.293, Springer, Berlin Heidelberg New York, 1987, p.p. 52–72.

[DGB] Desmedt Y., Goutier C. and Bengio S. Special uses and abuses of the Fiat-Shamir passport protocol. Lecture Notes in Computer Science, vol.293, Springer, Berlin Heidelberg New York, 1987, p.p. 21–39.

[DH] Diffie W. and Hellman M. New directions in cryptography. IEEE Transactions on Information Theory IT-22, 1976, p.p. 644–645.

[ElG] Gamal T.El. A public key cryptosystem and signature scheme based on discrete logarithms. IEEE Transactions on Information Theory IT-31, 1985, p.p. 469–473.

[EvY] Even S. and Yacobi Y. Cryptosystems which are NP-hard to break. Technion, Computer Science Department Technical Report, 1979.

[FFS] Feige U., Fiat A. and Shamir A. Zero knowledge proofs of idetity. Journal of Cryptology 1, 1988, p.p. 77–94.

[FiS] Fiat A. and Shamir A. How to prove yourself: practical solutions to identification and signature problems. Lecture Notes in Computer Science, vol.263, Springer, Berlin Heidelberg New York, 1987, p.p. 186–194.

[Fl] Floyd D. Annotated bibliography in conventional and public key cryptography. Cryptologia 7, 1983, p.p. 12–24.

[Ga] Gaines H.F. Cryptoanalysis. Dover Publications, New York, 1939.

[GMW] Goldreich O., Micali S. and Widgerson A. How to prove all NP-statements in zero-knowledge,and a methodology of cryptographic protocol design. Lecture Notes in Computer Science, vol.263, Springer, Berlin Heidelberg New York, 1987, p.p. 171–185.

[GM] Golgwasser S. and Micali S. Probabilistic encryption. Journal of Computer and System Sciences 28, 1984, p.p. 270–299.

[GMR] Golgwasser S., Micali S. and Rackoff C. The knowledge complexity of interactive proof systems. Proceedings of the 17th ACM Symposium on the Theory of Computing, 1985, p.p. 291–304.

[GMT] Goldwasser S., Micali S. and Tong P. Why and how to establish a private code on a public network. Proceedings of the 23rd FOCS Symposium, 1982, p.p. 134–144.

[Hil] Hill L.S. Cryptography in an algebraic alphabet. American Mathematical Monthly 36, 1929, p.p. 306–312.

[Ka] Kahn D. The codebreakers: the story of secret writing. Macmillan, New York, 1967.

[Kar1] Kari J. A cryptosystem based on propositional logic. Lecture Notes in Computer Science, vol.381, Springer, Berlin Heidelberg New York, 1989, p.p. 210–219.

[Kar2] Kari J. A cryptanalytic observations concerning systems based on language theory. Discrete Applied Mathematics 21, 1988, p.p. 265–268.

[Kar3] Kari J. Observations concerning a public-key cryptosystem based on interated morphisms. Theoretical Computer Science 66, 1989, p.p. 45–53.

[Ko] Koblitz N. A course in number theory and cryptography. Springer, Berlin Heidelberg New York, 1987.

[Kon] Konnheim A. Cryptography: a primer. Wiley and Sons, New York, 1982.

[Kra] Kranakis E. Primality and cryptography. Wiley-Teubner, Chichester New York Stuttgart, 1986.

[McE] McEliece R.J. A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report, Jet Propulsion Labs, Pasadena 42–44, 1978, p.p. 114–116.

[MeH] Merkle R. and Hellman M. Hiding information and signatures in trapdoor knapsacks. IEEE Transactions on Information Theory IT-24, 1978, p.p. 525–530.

[Mig] Mignotte M. How to share a secret. Lecture Notes in Computer Science, vol.149, Springer, Berlin Heidelberg New York, 1983, p.p. 371–375.

[Mil] Miller G.L. Riemann's hypothesis and tests for primality. Journal of Computer and System Sciences 13, 1976, p.p. 300–317.

[Nie] Niemi V. Hiding regular languages: a public-key cryptosystem. Manuscript, 1989.

[NuS] Nurmi H. and Salomaa A. On the cryptography of secret ballot. Behavioral Science, forthcoming.

[Odl] Odlyzko A.M. Discrete logarithms in finite fields and their cryptographic significance. Lecture Notes in Computer Science, vol.209, Springer, Berlin Heidelberg New York, 1985, p.p. 224-314.

[PoH] Pohlig S.C. and Hellman M. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. IEEE Transactions on Information Theory IT-24, 1978, p.p. 106–110.

[Rab1] Rabin M.O. Digitalized signatures and public key fuctions as intractable as factorization. MIT Laboratory for Computer Science Techical Report 212, 1979.

[Rab2] Rabin M.O. How to exchange secrets by oblivious transfer. Aiken Computation Laboratory, Harvard University, Techical Report TR–81, 1981.

[Ren] Renji Tao. Some results on the structure of feedforward inverses. Scientia Sinica, Ser.A 27, 1984, p.p. 157–162.

[RSA] Rivest M., Shamir A. and Adleman L. A method for obtaining digital signatures and public-key cryptosystems. ACM Communications 21, 1978, p.p. 120–126.

[RS] Rozenberg G. and Salomaa A. The mathematical theory of L systems. Academic Press, New York, 1980.

[Sa1] Salomaa A. Computation and automata. Cambridge University Press, Cambridge London New York, 1985.

[Sa2] Salomaa A. A public-key cryptosystem based on language theory. Computers and Security 7, 1988, p.p. 83–87.

[Sa3] Salomaa A. A deterministic algorithm for modular knapsack problems. Theoretical computer science (to appear).

[Sa4] Salomaa A. Decision problems arising from knapsack transformations. Acta cybernetica (to appear).

[SaY] Salomaa A. and Yu S. On a public-key cryptosystem based on interated morphisms and substitutions. Theoretical Computer Science 48, 1986, p.p. 283–296.

[SchA] Schnorr C.P. and Alexi W. RSA-bits are $0.5+\epsilon$ secure. Lecture Notes in Computer Science, vol.209, Springer, Berlin Heidelberg New York, 1985, p.p. 113–126.

[SP] Seberry J. and Pieprzyk J. Cryptography: an introduction to computer security. Prentice Hall, New York London, 1989.

[Sh1] Shamir A. A fast signature scheme. MIT Laboratory for Computer Science Technical Report, 1978.

[Sh2] Shamir A. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. Proceedings of the 23rd FOCS Symposium, 1982, p.p. 145–152.

[Sh3] Shamir A. Embedding cryptographic trapdoors in arbitrary knapsack systems. MIT Laboratory for Computer Science Technical Report 230, 1982.

[Sh4] Shamir A. Identity based cryptosystems and signature scheme. Lecture Notes in Computer Science, vol.196, Springer, Berlin Heidelberg New York, 1985, p.p. 47–53.

[Sh5] Shamir A. An efficient identification scheme based on permuted kernels. Weizmann Institute, Department of Applied Mathematics Technical Report, 1989.

[ShRA] Shamir A., Rivest R. and Adleman L. Mental Poker. In D.A.Klarner (ed.), The Mathematical gardner. Wadsworth International, Belmont, 1981, p.p. 37–43.

[Shan] Shannon C.E. Communication theory of secrecy systems. Bell System Technical Journal 28, 1949, p.p. 656–715.

[SiS] Siromoney R. and Siromoney G. A public key cryptosystem that defies cryptoanalysis. EATCS Bulletin 28, 1986, p.p. 37–43.

[Til] Van Tilbord H.C.A. An introduction to cryptology. Kluwer Academic Publishers, Boston Dordrecht Lancaster, 1988.

[WaM] Wagner N.R. and Magyarik M.R. A public key cryptosystem based on the word problem. Lecture Notes in Computer Science, vol.196, Springer, Berlin Heidelberg New York, 1985, p.p. 19–37.

[Wel] Welsh D. Codes and cryptography. Oxford University Press, Oxford, 1988.

[Wie] Wiener M.J. Cryptoanalysis of short RSA secret exponents. EUROCRYPT–89 Proceedings (to appear).

[Wil] Williams H.C. Some public-key crypto-functions as intracable as factorization. Cryptologia 9, 1985, p.p. 223–237.

[Yao] Yao A.C. Protocols for secure computations. Proceedings of the 23rd FOCS Symposium, 1982, p.p. 160–164.

[Zim] Zim H.S. Codes and secret writing. Scholastic Book Services, New York Toronto, 1948.

Предметный указатель

Оглавление