Глава 8

Шифрование — асимметричные методы

8.1 Введение

Стойкость классических шифров (например, шифра Цезаря) обеспечивалась абсолютной секретностью процесса шифрования. Современные шифры, такие как DES и AES, основаны на принципе Керхофса (см. раздел 7.1): алгоритмическое устройство этих шифров является открытым и доступным для исследования. Тем самым разработчики этих шифров стремились продемонстрировать, что стойкость их криптосистем зависит исключительно от выбора секретных ключей шифрования.

Существуют дополнительные возможности применения принципа Керхофса для уменьшения доли секретных компонентов алгоритма шифрования. Рассмотрим семантическое свойство шифрования, сформулированное Шенноном: преобразование перемешивания, абсолютно равномерно распределяющее осмысленные сообщения, принадлежащие области исходных текстов $\mathcal{M}$, по всему пространству сообщений $\mathcal{C}$ [264]. Как известно, такое распределение можно выполнить, не используя никаких секретов. Этот принцип был впервые реализован в работе Диффи (Diffie) и Хеллмана (Hellmann) в 1975 г. [97] (эта работа была опубликована в 1976 г., однако распространялась как препринт, начиная с декабря 1975 г. [96]). Они назвали свое изобретение **криптографией с открытым ключом** (public-key cryptography). В то время она представляла собой революционное открытие.

В криптосистемах с открытым ключом для шифрования не используется секретный ключ — он необходим только при расшифровке. В работе [97] Диффи и Хеллман кратко описали несколько математических преобразований, названных ими **однонаправленными функциями с секретом** (one-way trapdoor function). Они предназначались для реализации криптографии с открытым ключом и обладали следующими свойствами.

Свойство 8.1 (Однонаправленная функция с секретом). Однонаправленную функцию с секретом $f_t(x) : \mathcal{D} \mapsto \mathcal{R}$ легко вычислить для всех $x \in \mathcal{D}$, но очень трудно инвертировать для почти всех значений из $\mathcal{R}$. Однако, если используется

секретная информация t, то для всех значений $y \in \mathcal{R}$ легко вычислить величину $x \in \mathcal{D}$, удовлетворяющую условию $y = f_t(x)$.

Понятие однонаправленной функции с секретом — основное в криптографии с открытым ключом. Криптосистемы с открытым ключом, использующие однонаправленные функции с секретом, называются **асимметричными** (asymmetric cryptosystems). В первой работе Диффи и Хеллмана, посвященной криптографии с открытым ключом [97], было рассмотрено несколько однонаправленных функций с секретом. Однако эти функции были недостаточно асимметричными, поэтому вскоре Диффи и Хеллман предложили более удачный вариант: возведение в степень по модулю. Эта функция была использована в знаменитом криптографическом протоколе — протоколе обмена ключами Диффи–Хеллмана (Diffie-Hellman key exchange protocol) [98] (см. раздел 8.3). Эта первая удачная реализация криптографического алгоритма с открытым ключом широко используется до сих пор.

В 1974 году Меркл (Merkle) изобрел механизм согласования криптографического ключа путем явных асимметричных вычислений, получивших название *головоломка Меркла* [199]. Асимметричность головоломки Меркла заключается в том, что ее вычислительная сложность для законных участников протокола согласования ключа и для перехватчиков совершенно разная: легальные участники легко выполняют вычисления, а нелегальные — нет. Головоломка Меркла представляет собой первую эффективную реализацию однонаправленной функции с секретом. Несмотря на то что головоломка Меркла не подходит для применения в современных криптографических приложениях (мера ее асимметрии колеблется от n до n^2), ее влияние на криптосистемы с открытым ключом невозможно переоценить.

В настоящее время стало известно, что Кокс (Cocks), британский криптограф, изобрел первую криптосистему с открытым ключом в 1973 году [277]. Алгоритм шифрования Кокса, получивший название *алгоритма с несекретным ключом шифрования*, использует сложность разложения целого числа на простые множители и, по существу, совпадает с криптосистемой RSA (раздел 8.5). К сожалению, алгоритм Кокса был засекречен. В декабре 1997 года Группа по электронной защите средств связи (Communications Services Electronic Security Group — CESG) рассекретила алгоритм Кокса.

Несмотря на то что вначале криптосистемы с открытым ключом были достоянием узкого круга лиц, именно благодаря открытым исследованиям они нашли два важнейших применения: 1) цифровые подписи (раздел 10.4) и 2) обмен секретными ключами через открытые каналы связи (раздел 8.3). Эти два приложения лежат в основе электронной коммерции, осуществляемой через Internet.

8.1.1 Структурная схема главы

В начале главы вводится понятие стойкости “учебной криптографии”, которое сопровождается предупреждением о ее практической уязвимости (раздел 8.2). Затем описываются основные примитивы криптографии с открытым ключом: протокол обмена ключами Диффи–Хеллмана (раздел 8.3), учебный вариант алгоритма RSA (раздел 8.5), а также криптосистемы Рабина (раздел 8.10) и Эль-Гамаля (раздел 8.12). Эти алгоритмы и протоколы сопровождаются теоремами об их формальной стойкости при соответствующих предположениях. В частности, формулируется задача Диффи–Хеллмана и задача дискретного логарифмирования (раздел 8.4), задача RSA (раздел 8.7) и задача о разложении целого числа на простые множители (раздел 8.8). Кроме того, в главе вводятся формальные понятия для описания разнообразных атак на криптосистемы с открытым ключом (раздел 8.6). Нестойкость учебных вариантов криптографических алгоритмов демонстрируется на примере алгоритма RSA (раздел 8.9), Рабина (раздел 8.11) и Эль-Гамаля (раздел 8.13). В разделе 8.14 рассматривается необходимость более строгого понятия стойкости шифрования с открытым ключом. В разделе 8.15 описывается комбинация симметричных и асимметричных криптосистем — гибридная криптосистема.

8.2 Нестойкость “учебных” алгоритмов шифрования

Следует отметить, что алгоритмы шифрования, описанные в этой главе, носят учебный характер. Их можно найти в любом учебнике по криптографии. К сожалению, эти алгоритмы непригодны для практического применения. Стойкость “учебного” алгоритма шифрования в криптосистеме с открытым ключом описывается следующим свойством.

Свойство 8.2 (Нестойкость “учебных” алгоритмов шифрования). В рамках этой главы стойкость (секретность) криптосистемы рассматривается с двух точек зрения.

1. **Стойкость по принципу “все или ничего”.** Имея текст, зашифрованный определенным алгоритмом, атакующий должен восстановить блок исходного текста, размер которого, как правило, определяется параметром безопасности криптосистемы. Имея исходный и зашифрованный текст, атакующий должен восстановить целый блок секретного ключа. При этом атакующий либо добивается полного успеха, либо не получает ничего. Следует обратить особое внимание на слово “ничего”: оно означает, что атакующий не имеет никакой секретной информации ни до, ни после безуспешной атаки.
2. **Пассивная атака.** Атакующий не манипулирует зашифрованным текстом и не модифицирует его, используя имеющиеся данные. Кроме того, он не

обращается к владельцу ключа с просьбой расшифровать или зашифровать сообщение.

Понятие стойкости, сформулированное выше, чрезвычайно расплывчато и непригодно для применения на практике. Точнее было бы назвать его "нестойкостью".

Поясним, почему свойство 8.2.1 на самом деле описывает нестойкость алгоритма. В приложениях исходные данные, как правило, содержат часть несекретной информации, которая может быть известной атакующему. Например, некоторые данные имеют небольшой диапазон изменения: зарплата не может превышать один миллион — числа, весьма небольшого по криптографическим меркам. В качестве второго примера можно указать пароли, состоящие из восьми символов. Довольно часто дополнительная информация позволяет атакующему распознать весь исходный текст.

Теперь покажем, почему свойство 8.2.2 также определяет нестойкость алгоритма. Никогда не следует полагаться на то, что атакующий инертен и пассивен. Как правило, атакующие не стесняются при выборе средств. В частности, они вступают в контакт с атакованным пользователем, посылают ему зашифрованный текст для последующей расшифровки и требуют вернуть исходный текст. Такой способ общения с пользователем (владельцем открытого ключа) предоставляет атакующему **услуги оракула шифрования**. Как будет показано в дальнейшем, избежать этого очень трудно.

Замечательные алгебраические свойства "учебных" криптографических алгоритмов часто позволяют атакующему, прибегающему к услугам оракула, взламывать криптосистему. В главе будет продемонстрировано несколько случаев успешных атак, а в последующих главах показана эффективность таких приемов.

Глава содержит несколько предупреждений об опасности услуг оракула. Следует отметить, что обычные пользователи алгоритмов с открытым ключом часто весьма наивны и предоставляют такие услуги атакующим. Кроме того, избежать этого очень трудно (см. раздел 12.5.4). Правильная стратегия заключается в том, что криптосистема должна оставаться стойкой, даже если с ней работают неопытные пользователи.

Сформулировав свойство 8.2, мы явно указали, что в рамках этой главы не рассматривается строгое понятие стойкости алгоритмов с открытым ключом. Следовательно, "учебные" алгоритмы шифрования не являются стойкими в строгом смысле этого слова. Наоборот, в главе показано, что "учебные" алгоритмы шифрования весьма слабы, поскольку допускают утечку информации. Способы исправления этих недостатков в главе не описываются.

Более строгие определения стойкости, позволяющие противостоять более сильным (т.е. реальным) атакам, будут даны в главе 14. Практические аналоги "учебных" алгоритмов шифрования описываются в главе 15.

8.3 Протокол обмена ключами Диффи–Хеллмана

В симметричных криптосистемах перед началом работы необходимо передать секретный ключ обеим сторонам. До появления криптосистем с открытым ключом распределение секретных ключей между общающимися сторонами всегда представляло собой сложную задачу, поскольку для этого необходим защищенный канал. Как правило, для обмена ключами использовался специальный курьер. Важным преимуществом криптографии с открытым ключом над симметричными криптосистемами является обмен ключами между удаленными пользователями без применения защищенного канала. Первая практичная схема такого обмена была предложена Диффи и Хеллманом и стала называться протоколом обмена экспоненциальными ключами Диффи–Хеллмана [98].

Для начала пользователи, Алиса и Боб, договариваются использовать конечное поле $\mathbb{F}_q$ и элемент $g \in \mathbb{F}_q$, порождающий группу, имеющую большой порядок. Для простоты будем рассматривать поле $\mathbb{F}_p$, в котором число p является большим простым числом. Стороны могут проверить простоту числа p, используя алгоритм 4.5, позволяющий создать число p, для которого известно полное разложение числа $p-1$ на множители. Затем, используя алгоритм 5.1, Алиса и Боб могут найти элемент g, порождающий группу $\mathbb{F}_p^*$. По теореме 5.11 каждое число из интервала $[1, p)$ можно представить в виде $g^x \pmod p$, где x — некоторое число. Теперь числа p и q можно использовать в качестве общих исходных данных в основном варианте протокола обмена ключами Диффи–Хеллмана.

Протокол 8.1. Протокол обмена ключами Диффи–Хеллмана

ОБЩИЕ ИСХОДНЫЕ ДАННЫЕ:

$(p, g) : p$ — большое простое число,
g — порождающий элемент группы $\mathbb{F}_p^*$.

РЕЗУЛЬТАТ:

Элемент группы $\mathbb{F}_p^*$, разделенный между Алисой и Бобом.

1. Алиса генерирует элемент $a \in_U [1, p-1)$, вычисляет число $g_a \leftarrow g^a \pmod p$ и посылает его Бобу.
2. Боб генерирует элемент $b \in_U [1, p-1)$, вычисляет число $g_b \leftarrow g^b \pmod p$ и посылает его Алисе.
3. Алиса вычисляет значение $k \leftarrow g_b^a \pmod p$.
4. Боб вычисляет число $k \leftarrow g_a^b \pmod p$.

Из протокола 8.1 легко видеть, что значение, вычисляемое Алисой, равно

$$k = g^{ba} (\mathrm{mod}\, p),$$

а значение, вычисляемое Бобом, —

$$k = g^{ab} (\mathrm{mod}\, p).$$

Заметим, что, поскольку $ab \equiv ba (\mathrm{mod}\, p-1)$, обе стороны вычисляют одно и то же значение. Это обеспечивает разделение ключа между двумя сторонами.

Таким образом, числа p и g можно разослать всем участникам системы.

Пример 8.1. Допустим, что $p = 43$. Применяя алгоритм 5.1, находим, что число 3 является первообразным корнем по модулю 43. Распределим между Алисой и Бобом открытую пару $(p, g) = (43, 3)$. Для того чтобы обменяться секретными ключами, Алиса генерирует случайную секретную степень, равную 8, и посылает Бобу число $3^8 \equiv 25 (\mathrm{mod}\, 43)$. Боб генерирует секретное число 37 и посылает Алисе число $3^{37} \equiv 20 (\mathrm{mod}\, 43)$. Секретный ключ, согласованный Алисой и Бобом, равен

$$9 \equiv 20^8 \equiv 25^{37} (\mathrm{mod}\, 43).$$

□

Описание протокола Диффи–Хеллмана следует дополнить следующими замечаниями.

- Простое число p следует выбирать так, чтобы число $p-1$ имело достаточно большой простой множитель p'. Слова "достаточно большой" означают, что $p' > 2^{160}$. Необходимость этого условия обсуждается в разделе 8.4.
- Число g не обязано быть порождающим элементом группы $\mathbb{F}_p^*$. Необходимо лишь, чтобы оно было порождающим элементом ее подгруппы, имеющей достаточно большой порядок, например, подгруппы порядка p'. В этом случае Алиса и Боб должны проверить условия $g \neq 1$ и $g^{p'} \equiv 1 (\mathrm{mod}\, p)$. По этой причине число p' должно быть частью общих исходных данных.
- Алиса (соответственно Боб) должна проверить условие $g_b \neq 1$ (соответственно $g_a \neq 1$). Это условие гарантирует, что для выбранной ею степени, принадлежащей интервалу $(1, p')$, разделенный ключ g^{ab} будет элементом подгруппы порядка p' группы $\mathbb{F}_p$, т.е. элементом достаточно большой подгруппы.
- По окончании протокола Алиса (соответственно Боб) должна стереть свою степень a (соответственно степень b). Этим они обеспечивают **заблаговременную секретность** (forward secrecy) ключа g^{ab}. Это свойство обсуждается в разделах 8.15 и 11.6.1.

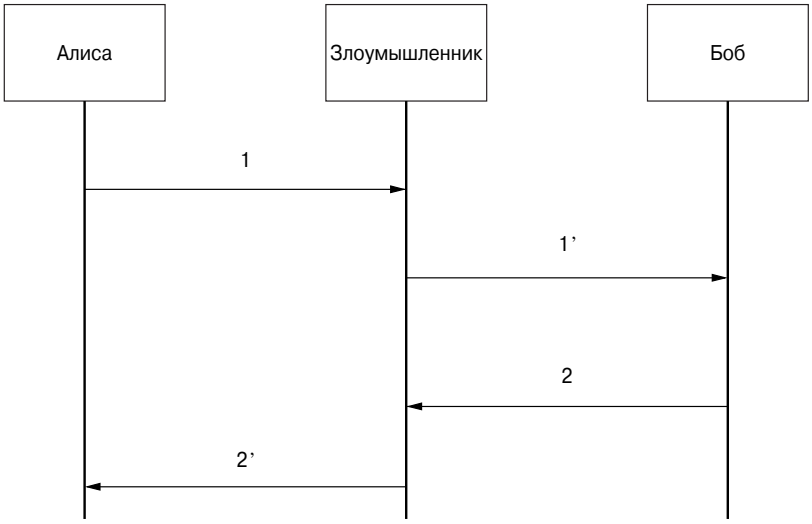
8.3.1 Атака “человек посередине”

Следует иметь в виду, что протокол обмена ключами Диффи–Хеллмана не обеспечивает аутентичности согласованного ключа. Активный противник, внед-

Атака 8.1. Атака “человек посередине” на протокол обмена ключами Диффи–Хеллмана

ОБЩИЕ ИСХОДНЫЕ ДАННЫЕ:

Как в протоколе 8.1.



1. Алиса генерирует элемент $a \in_U [1, p-1)$, вычисляет значение $g_a \leftarrow g^a \pmod p$ и посылает его Злоумышленнику (“Бобу”).

1’. Злоумышленник (“Боб”) вычисляет значение $g_m \leftarrow g^m \pmod p$, где $m \in \in [1, p-1)$ и посылает его Бобу

2. Боб генерирует элемент $b \in_U [1, p-1)$, вычисляет значение $g_b \leftarrow g^b \pmod p$ и посылает его Злоумышленнику (“Алисе”).

2’. Злоумышленник (“Алиса”) посылает Алисе число g_m.

3. Алиса вычисляет значение $k_1 \leftarrow g_m^b \pmod p$. (* Этот ключ распределен между Алисой и Злоумышленником, поскольку Злоумышленник может вычислить значение $k_1 \leftarrow g_a^m \pmod p$. *)

4. Боб вычисляет значение $k_2 \leftarrow g_m^b \pmod p$. (* Этот ключ распределен между Бобом и Злоумышленником, поскольку Злоумышленник может вычислить значение $k_2 \leftarrow g_m^b \pmod p$. *)

ренный в канал связи между Алисой и Бобом, может манипулировать сообщениями протокола и начать **атаку "человек посередине"** (man-in-the-middle attack).

В ходе этой атаки Злоумышленник перехватывает и блокирует первое сообщение Алисы Бобу, т.е. число g_a, маскируется под Алису и посылает Бобу следующее сообщение.

Злоумышленник ("Алиса") — Бобу: $g_m \left(\stackrel{\text{def}}{=} g^m\right) (\mathrm{mod}\, p)$.

(Напоминаем соглашения, принятые в разделе 2.6.2.) Боб, следуя инструкциям протокола, возвращает Злоумышленнику ("Алисе") число g_b. Это число снова перехватывается и блокируется Злоумышленником. Теперь Злоумышленник и Боб согласовали между собой ключ $g^{bm} (\mathrm{mod}\, p)$, хотя Боб считает, что он разделил этот ключ с Алисой.

Аналогично Злоумышленник, имитируя Боба, может согласовать другой ключ с Алисой (т.е. число $g^{am} (\mathrm{mod}\, p)$). Впоследствии Злоумышленник может использовать оба ключа для чтения и подмены "секретных" сообщений, которыми обмениваются Алиса и Боб, или поочередно имитировать этих пользователей.

Атака на протокол обмена ключами Диффи–Хеллмана вполне реальна, поскольку этот протокол не предусматривает проверки аутентичности источника сообщений. Для того чтобы ключи были согласованы только между Алисой и Бобом, обе стороны должны быть уверены друг в друге. Способы аутентификации рассматриваются в главе 11. В этой же главе (раздел 11.6) описан метод безопасного применения протокола обмена ключами Диффи–Хеллмана.

8.4 Задача Диффи–Хеллмана и задача дискретного логарифмирования

Стойкость разделенного ключа в протоколе Диффи–Хеллмана обеспечивается вычислением значениye $g^{ab} (\mathrm{mod}\, p)$ по заданным числам g_a и g_b. Эта задача называется **вычислительной проблемой Диффи–Хеллмана** (CDH problem — Diffie-Hellman problem).

Определение 8.1 (Вычислительная проблема Диффи–Хеллмана) *(в конечном поле).*

ИСХОДНЫЕ ДАННЫЕ:

desc$(\mathbb{F}_q)$*: описание конечного поля* $\mathbb{F}_q$*;*

$g \in \mathbb{F}_q^*$ *— порождающий элемент группы* $\mathbb{F}_q^*$*;*

$g_a, g_b \in \mathbb{F}_q^*$*, где* $0 < a, b < q$.

РЕЗУЛЬТАТ:

g^{ab}.

Эта формулировка представляет собой общую постановку задачи в конечном поле $\mathbb{F}_q$. В протоколе обмена ключами Диффи–Хеллмана используется более специальный случай. Общая постановка приводится только для формализации задачи, хотя для большей ясности мы будем изучать лишь частный случай.

Если бы проблему Диффи–Хеллмана было легко решить, то значение $g^{ab} (\mathrm{mod}\, p)$ можно было бы найти, зная числа p, g, g_a и g_b, которые являются частью протокола. Предполагая, что противник обладает возможностями перехвата информации (раздел 2.3), следует предположить, что эти числа не являются для него секретом.

Проблема Диффи–Хеллмана опирается на сложность вычисления **дискретного логарифма** (discrete logarithm problem).

Определение 8.2 (Задача дискретного логарифмирования) *(в конечном поле).*
ИСХОДНЫЕ ДАННЫЕ:

desc$(\mathbb{F}_q)$*: описание конечного поля* $\mathbb{F}_q$*;*
$g \in \mathbb{F}_q^*$ *— порождающий элемент группы* $\mathbb{F}_q^*$*;*
$h \in_U \mathbb{F}_q^*$.

РЕЗУЛЬТАТ:

Уникальное число $a < q$*, удовлетворяющее условию* $h = g^a$.

Целое число a *обозначается как* $\log_g h$.

Дискретное логарифмирование аналогично обычному логарифмированию в поле действительных чисел. Однако в отличие от последней задачи, в которой решение является приближенным, задача о вычислении дискретного логарифма имеет *точное* решение.

Как указывалось в главе 4, в основе современной криптографии лежит теория вычислительной сложности. Это значит, что стойкость криптосистем с открытым ключом является *условной* и зависит от сложности решения некоторых задач. Проблема Диффи–Хеллмана и задача дискретного логарифмирования считаются трудноразрешимыми. Интуитивно ясно, что сложность решения этих задач зависит как от размера поля $\mathbb{F}_q$, так и от выбора параметров (открытого параметра g и секретных чисел a и b). Очевидно, что небольшие варианты этих задач являются разрешимыми. В свое время мы покажем, что существуют другие варианты этих задач, которые относительно легко решить. Таким образом, точное описание понятия сложности решения задачи должно учитывать как размер задачи, так и выбор ее варианта. Сведения из теории сложности, приведенные в главе 4, позволяют сформулировать точные предположения о неразрешимости проблемы Диффи–Хеллмана и задачи дискретного логарифмирования. Рекомендуем читателям вспомнить некоторые понятия и обозначения, используемые в теории вычислительной сложности (в частности, “1^k”, “вероятностный полиномиальный алгоритм” и “пренебрежимо малая величина”).

Предположение 8.1 (Условия неразрешимости проблемы Диффи–Хеллмана). *Алгоритмом решения проблемы Диффи–Хеллмана называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[g^{ab} \leftarrow \mathcal{A}\left(\mathrm{desc}\left(\mathbb{F}_q\right), g, g^a, g^b\right)\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.1.

Пусть $\mathcal{IG}$ — генератор вариантов, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — 1) $\mathrm{desc}(\mathbb{F}_q)$, где $|q| = k$, и 2) порождающий элемент $g \in \mathbb{F}_q^*$.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости проблемы Диффи–Хеллмана, если для вариантов $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по сравнению со всеми достаточно большими числами k.

Предположение 8.2 (Условия неразрешимости задачи дискретного логарифмирования). *Алгоритмом решения задачи дискретного логарифмирования называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[\log_g h \leftarrow \mathcal{A}\left(\mathrm{desc}\left(\mathbb{F}_q\right), g, h\right)\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.2.

Пусть $\mathcal{IG}$ — генератор вариантов, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — 1) $\mathrm{desc}(\mathbb{F}_q)$, где $|q| = k$, 2) порождающий элемент $g \in \mathbb{F}_q^*$ и 3) элемент $h \in \mathbb{F}_q^*$.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости задачи дискретного логарифмирования, если для вариантов $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по отношению ко всем достаточно большим числам k.

Иначе говоря, здесь предполагается, что почти все достаточно большие варианты указанных задач в конечных полях не имеют эффективного алгоритма решения. Доля слабых вариантов этих задач, поддающихся решению, пренебрежимо мала.

И все же эти два предположения требуют уточнения. Для начала сделаем несколько формальных утверждений.

Примечание 8.1.

1. *В предположениях 8.1 и 8.2 соответствующие вероятностные пространства должны учитывать 1) пространство вариантов, из которых извлекаются произвольные конечные поля и элементы (важность этого условия обсуждается в разделе 8.4.1) и 2) пространство случайных операций в эффективном алгоритме. Второе условие необходимо, поскольку в множество*

полиномиальных, или эффективных, алгоритмов входят и рандомизированные алгоритмы (см. определение 4.6 в разделе 4.4.6).

2. *Число k в обоих формулировках называется* ***параметром безопасности*** *(security parameter), а $\mathcal{IG}(1^k)$ — случайный вариант конечного поля и элементов. Изучая вероятностное генерирование простых чисел (раздел 4.4.6.1) и конструкцию поля (раздел 5.4), мы убедились, что вариант $\mathcal{IG}(1^k)$ действительно можно решить за время, полиномиально зависящее от числа k. Считается общепризнанным, что число $k = 1024$ является нижней границей параметра безопасности для задачи дискретного логарифмирования в конечных полях. Эта нижняя граница установлена, исходя из существования субэкспоненциального алгоритма решения этой задачи (индексное исчисление). Определением субэкспоненциальной сложности является выражение (8.4.2). Для числа $|q| = 1024$ это выражение равно 2^{80}. По этой причине в качестве нижней границы используется число $k = 1024$. Таким образом, выражение "для всех достаточно больших чисел k" означает, что числа k больше* 1024.

3. *Предположение о невозможности дискретного логарифмирования означает, что функция*

$$g^x : \mathbb{Z}_q \mapsto \mathbb{F}_q^* \tag{8.4.1}$$

является однонаправленной. Считается, что это условие действительно выполняется (описание гипотезы $\mathcal{P} \neq \mathcal{NP}$ см. в разделе 4.5), т.е. однонаправленная функция существует.

4. *В настоящее время неизвестно, является ли функция (8.4.1) функцией с секретом (см. свойство 8.1 в разделе 8.1). Иначе говоря, никому не удалось внедрить в эту функцию секретную информацию, позволяющую эффективно вычислить обратную функцию (т.е. разработать эффективный метод, позволяющий по числу g^x найти число x, используя секретную информацию). Однако, если эта функция использует составные модули (оставаясь однонаправленной), она становится функцией с секретом, в которой роль секрета играет разложение модуля на простые множители. Технические детали читатели найдут в книгах [224, 228, 229].* □

Сформулированные выше предположения, по сути, означают, что "не существует полиномиальных по параметру k алгоритмов решения этих задач". Однако к этому утверждению следует относиться с большой осторожностью. Если полиномиальный алгоритм $\mathrm{poly}(k)$ существует, его время работы не превышает k^n, где n — некоторое целое число. С другой стороны, известно, что существует субэкспоненциальный алгоритм дискретного логарифмирования, время работы которого определяется выражением

$$\mathrm{sub\_exp}(q) = \exp(c(\log q)^{1/3}(\log\log q)^{2/3}), \tag{8.4.2}$$

где c — небольшая константа (например, $c < 2$). Сравнивая выражения "не существует полиномиального алгоритма дискретного логарифмирования $\mathrm{poly}(k)$" и "существует субэкспоненциальный алгоритм дискретного логарфмирования $\mathrm{sub\_exp}(q)$", приходим к выводу, что число k^n чрезвычайно мало по сравнению с числом $\mathrm{sub\_exp}(k \log 2)$ (если $k = |q| = \log_2 q$, то $q = k \log 2$). Однако выражение "чрезвычайно мало" оказывается истинным, только если число n фиксировано, а число k (будучи функцией числа n) — достаточно велико. Разберемся в этом подробнее.

Допустим, что число k не является достаточно большим. Возьмем натуральный логарифм от выражений $\mathrm{poly}(k)$ и $\mathrm{sub\_exp}(k \log 2)$ и сравним полученные результаты:

$$n(\log k)^{1/3} \text{ и } c' k^{1/3},$$

где $c' = c(\log 2)^{1/3} < c$. Теперь ясно, что, если число n сравнимо с числом $c' k^{1/3}$, субэкспоненциальный алгоритм дискретного логарифмирования работает быстрее, чем гипотетически "несуществующий полиномиальный алгоритм". Выражение "не существует полиномиального алгоритма дискретного логарифмирования $\mathrm{poly}(k)$" приобретает реальный смысл, только если число k не ограничено (и, следовательно, может быть "достаточно большим", если число n фиксировано). В действительности число k не может быть неограниченным. В частности, при $k = 1024$ (общепринятая нижняя граница параметра безопасности) и $c < 2$ существует полиномиальный алгоритм дискретного логарифмирования $\mathrm{poly}(k)$, время работы которого является полиномом 9-й степени от параметра k (см. пример 8.4).

До сих пор мы использовали *асимптотическую* интерпретацию выражения "не существует полиномиального алгоритма дискретного логарифмирования $\mathrm{poly}(k)$": число k считалось неограниченным и достаточно большим. В действительности число k *должно быть ограничено*, а значит, алгоритм $\mathrm{poly}(k)$ *существует*. Несмотря на это, можно установить настолько большую границу для числа k, что полиномиальный алгоритм будет работать немыслимо долго. Считается, что число $k = 1024$ вполне удовлетворяет этому условию.

В остальной части книги будет использоваться именно асимптотическое толкование выражения "не существует полиномиального алгоритма решения $\mathrm{poly}(k)$".

В заключение рассмотрим взаимосвязь между вычислительной проблемой Диффи–Хеллмана и задачей дискретного логарифмирования.

Заметим, что доступность чисел $a = \log_g g_1$ или $b = \log_g g_2$ позволяет найти числа

$$g^{ab} = g_1^b = g_2^b.$$

Иначе говоря, если существует эффективный алгоритм, решающий вычислительную проблему Диффи–Хеллмана, значит, существует эффективный алгоритм дискретного логарифмирования. Следовательно, если не выполняются условия невоз-

можности дискретного логарифмирования, то не выполняются и условия неразрешимости вычислительной проблемы Диффи–Хеллмана. Таким образом, вычислительная проблема Диффи–Хеллмана проще задачи дискретного логарифмирования, т.е. условия неразрешимости вычислительной проблемы Диффи–Хеллмана строже, чем условия неразрешимости задачи дискретного логарифмирования. Обратное утверждение еще не доказано.

Могут ли выполняться условия невозможности дискретного логарифмирования, если условия о неразрешимости вычислительной проблемы Диффи–Хеллмана не выполняются?

Маурер (Maurer) и Вольф (Wolf) привели убедительные эвристические аргументы, обосновывающие взаимозависимость этих задач. Они считают весьма вероятным, что эти две задачи эквивалентны [190].

8.4.1 Произвольность вариантов и условия неразрешимости

Следует подчеркнуть, что в условиях невозможности дискретного логарифмирования варианты должны быть произвольными. Рассмотрим группу $\mathbb{F}_p^*$, где p — простое число, состоящее из k бит, и попробуем найти число a, удовлетворяющее условию $h \equiv g^a (\mathrm{mod}\, p)$.

Известно, что число a является элементом поля $\mathbb{Z}_{p-1}$. Если $p-1 = q_1 q_2 \ldots q_\ell$, где каждый множитель q_i мал (т.е. q_i не превосходит полином от параметра k при $i = 1, 2, \ldots, \ell$), то задача дискретного логарифмирования сводится к вычислению значений $a_i \equiv a (\mathrm{mod}\, q_i)$ по числам $h^{(p-1)/q_i} (\mathrm{mod}\, p)$, только теперь числа a_i малы и могут быть найдены за полиномиальное время, зависящее от параметра k. После определения чисел $a_1, a_2, \ldots, a_\ell$ число a находится по китайской теореме об остатках (теорема 6.7). Эта идея лежит в основе полиномиального алгоритма Полига (Pohlig) и Хеллмана [231], позволяющего решить задачу дискретного логарифмирования при условии, что число $p-1$ не имеет большого простого множителя. Очевидно, что если каждый простой множитель числа $p-1$ ограничен полиномом, зависящим от параметра k, то время выполнения алгоритма Полига–Хеллмана полиномиально зависит от параметра k.

Простое число p, такое что число $p-1$ не имеет большого простого множителя, называется **гладким** (smooth prime). Правда, иногда в этом же смысле употребляется выражение “число $p-1$ является гладким”. Стандартный способ избежать гладких простых чисел заключается в конструировании такого простого числа p, чтобы $p-1$ делилось на другое большое простое число p'. По теореме 5.2.2 циклическая группа $\mathbb{F}_p^*$ содержит единственную подгруппу, имеющую порядок p'. Если число p' сделать открытым, пользователи протокола обмена ключами Диффи–Хеллмана могут быть уверенными в том, что этот протокол ра-

ботает в этой большой подгруппе. Все что нужно для этого — найти элемент $g \in \mathbb{F}_p^*$, удовлетворяющий условию

$$g^{(p-1)/p'} \not\equiv 1 (\mathrm{mod}\, p).$$

Элемент g порождает группу, имеющую простой порядок p'. В качестве общих исходных данных протокол обмена ключами Диффи–Хеллмана должен использовать числа (p, p', g), удовлетворяющие указанным выше свойствам. Считается, что размер простого числа p' должен быть не меньше 160 бит, т.е. $p' > 2^{160}$ (см. раздел 10.4.8.1).

Проблема Диффи–Хеллмана и задача дискретного логарифмирования считаются неразрешимыми в конечной абелевой группе большого порядка, например, в подгруппе конечного поля, имеющей большой простой порядок, или в группе точек эллиптической кривой, определенной над конечным полем (конструкция групп рассматривается в разделе 5.5, а задача дискретного логарифмирования на эллиптических кривых — в разделе 5.5.3). Таким образом, протокол обмена ключами Диффи–Хеллмана правильно работает в этих группах.

Существует несколько эффективных субэкспоненциальных алгоритмов дискретного логарифмирования при условии, что вычисляемое значение невелико — например, λ-метод Полларда, описанный в разделе 3.6.1. Вычисление небольших дискретных логарифмов применяется во многих криптографических протоколах.

Проблема Диффи–Хеллмана исследуется весьма активно. Большой список литературы, посвященной этой теме, содержится в обзоре Одлыжко (Odlyzko) [221].

8.5 Криптосистема RSA (учебный вариант)

Наиболее известной криптосистемой с открытым ключом является алгоритм RSA, названный по первым буквам фамилий своих изобретателей — Ривеста (Rivest), Шамира (Samir) и Адлемана (Adleman) [246]. Криптосистема RSA — первая практическая реализация криптографии с открытым ключом на основе понятия однонаправленной функции с секретом, предложенного Диффи и Хеллманом [97, 98].

Криптосистема RSA описывается алгоритмом 8.1. Отметим, что этот алгоритм носит учебный характер.

Покажем, что система, описанная алгоритмом 8.1, действительно является криптографической, т.е. процедура расшифровки, выполненная Алисой, действительно восстанавливает исходный текст, зашифрованный Бобом.

Из определения операций модулярной арифметики (определение 4.4 из раздела 4.3.2.5) следует, что сравнение $ed \equiv 1 (\mathrm{mod}\, \phi(N))$ в алгоритме 8.1 эквивалентно уравнению

$$ed = 1 + k\phi(N),$$

Алгоритм 8.1. Криптосистема RSA

Ключ

Для того чтобы установить ключ, Алисе необходимо выполнить следующие операции.

1. Выбрать два случайных простых числа p и q, удовлетворяющих условию $|p| \approx |q|$.
   (* Для этого можно воспользоваться методом Монте-Карло, описанным алгоритмом 4.7. *)
2. Вычислить $N = pq$.
3. Вычислить $\phi(N) = (p-1)(q-1)$.
4. Выбрать случайное целое число $e < \phi(N)$, удовлетворяющее условию $\gcd(e, \phi(N)) = 1$, и найти целое число d, такое что

$$ed \equiv 1 (\mathrm{mod}\, \phi(N)).$$

   (* Поскольку $\gcd(e, \phi(N)) = 1$, это уравнение имеет решение d, которое можно найти с помощью расширенного алгоритма Евклида (алгоритм 4.2). *)
5. Использовать пару (N, e) в качестве параметров открытого ключа, тщательно уничтожить числа p, q и $\phi(N)$ и запомнить число d в качестве закрытого ключа.

Шифрование

Для того чтобы переслать Алисе секретное сообщение, имеющее длину $m < N$, Боб создает зашифрованный текст c.

$$c \leftarrow m^e (\mathrm{mod}\, N).$$

(* С точки зрения Боба, пространство исходных сообщений представляет собой множество всех положительных чисел, которые меньше числа N, хотя на самом деле этим пространством является группа $\mathbb{Z}_N^*$. *)

Расшифровка

Для того чтобы расшифровать зашифрованный текст c, Алиса вычисляет формулу

$$m \leftarrow c^d (\mathrm{mod}\, N).$$

где k — некоторое целое число. Следовательно, значение, вычисленное Алисой в результате выполнения процедуры расшифровки, определяется по следующей

формуле.

$$c^d \equiv m^{ed} \equiv m^{1+k\phi(N)} \equiv m \cdot m^{k\phi(N)} (\mathrm{mod}\, N). \quad (8.5.1)$$

Следует отметить, что неравенство $m < N$ практически всегда означает, что $m \in \mathbb{Z}_N^*$ (т.е. почти все числа, которые меньше числа N, принадлежат мультипликативной группе целых чисел, взаимно простых с числом N). Условие $m \in \mathbb{Z}_N^*$ нарушается, если $m = up$ или $m = vq$, где $u < q$ и $v < p$. В этих ситуациях Боб может разложить число N на простые множители, вычислив значение $\gcd(m, N)$. Предполагая, что это является трудноразрешимой задачей (условия, при которых факторизация числа представляет собой трудноразрешимую задачу, будут сформулированы позднее), можно предположить, что любое сообщение $m < N$, созданное Бобом, удовлетворяет условию $m \in \mathbb{Z}_N^*$.

Если $m \in \mathbb{Z}_N^*$, то по теореме Лагранжа (следствие 5.2)

$$\mathrm{ord}_N(m) \mid \#\mathbb{Z}_N^* = \phi(N).$$

Это утверждение справедливо для всех $m \in \mathbb{Z}_N^*$. В соответствии с определением порядка элемента группы (определение 5.9 в разделе 5.2.2) это означает, что для всех $m \in \mathbb{Z}_N^*$ выполняется условие

$$m^{\phi(N)} \equiv 1 (\mathrm{mod}\, N).$$

Отсюда следует, что

$$m^{k\phi(N)} \equiv (m^{\phi(N)})^k \equiv 1 (\mathrm{mod}\, N)$$

для любого целого числа k. Итак, величина в формуле (8.5.1) действительно равна числу m.

Пример 8.2. Допустим, что Алиса выбрала числа $N = 7 \times 13 = 91$ и $e = 5$. Тогда $\phi(N) = 6 \times 12 = 72$. Применяя алгоритм 4.2 к паре $(a, b) = (72, 5)$, Алиса получает следующее число.

$$72 \times (-2) + 5 \times 29 = 1.$$

Иначе говоря, $5 \times 29 \equiv 1 (\mathrm{mod}\, 72)$. Следовательно, секретным показателем степени, используемым Алисой при шифровании, является число 29. Алиса устанавливает пару $(N, e) = (91, 5)$ в качестве параметров открытого ключа криптосистемы RSA.

Допустим, что Боб шифрует исходное сообщение $m = 3$, используя формулу

$$c = 3^5 = 243 \equiv 61 (\mathrm{mod}\, 91).$$

Зашифрованное сообщение представляет собой число 61.

Для того чтобы расшифровать сообщение 61, Алиса вычисляет значение

$$61^{29} \equiv 3 (\mathrm{mod}\, 91).$$

□

8.6 Взлом криптосистем с открытым ключом с помощью криптоанализа

Можно сказать, что "криптосистема X является стойкой по отношению к атаке Y, но нестойкой по отношению к атаке Z", т.е. стойкость криптосистемы зависит от разновидности атаки. Активные атаки можно разделить на три вида.

Определение 8.3 (Активные атаки на криптосистемы).

Атака на основе подобранного открытого текста *(chosen-plaintext attack — CPA). Атакующий выбирает исходные сообщения и передает их шифровальщику для получения зашифрованных текстов. Задача атакующего — взломать криптосистему, используя полученные пары открытых и зашифрованных текстов.*

Атака на основе подобранного зашифрованного текста *(chosen-ciphertext attack — CCA). Атакующий выбирает зашифрованные сообщения и передает их на расшифровку для получения исходных сообщений. Цель атакующего — взломать криптосистему, используя полученные пары открытых и зашифрованных текстов. Атакующий достигает успеха, если он раскрывает ключ и способен в дальнейшем извлекать секретную информацию из зашифрованного текста, не прибегая к посторонней помощи.*

Атака на основе адаптивно подобранного зашифрованного текста *(adaptive chosen-ciphertext attack — CCA2). Это — разновидность атаки CCA, в которой услуги расшифровки доступны для всех зашифрованных текстов, за исключением заданного.*

Эти атаки можно описать следующими сценариями.

- В атаке на основе подобранного открытого текста атакующий владеет блоком шифрования.
- В атаке на основе подобранного зашифрованного текста атакующий имеет ограниченный доступ к блоку расшифровки: после нескольких попыток доступ к блоку закрывается, и расшифровку требуемого текста атакующий должен выполнять без его помощи.
- В атаке на основе адаптивно подобранного зашифрованного текста атакующий владеет блоком расшифровки сколь угодно долго, однако, как и в предыдущем случае, расшифровка требуемого текста должна выполняться без его помощи. Это условие вполне естественно — иначе атакующему незачем взламывать криптосистему.

Все эти атаки основаны на предположении, что атакующему не известен криптографический ключ.

Атаки CPA и CCA изначально предназначались для активного криптоанализа криптосистем с секретным ключом. Целью этого криптоанализа являлся взлом криптосистемы с помощью пар открытых и зашифрованных сообщений, получаемых в ходе атаки [284]. Затем они были адаптированы для криптоанализа криптосистем с открытым ключом. Следует отметить три особенности криптосистем с открытым ключом.

- Услуги шифрования в криптосистеме с открытым ключом доступны любому желающему, поскольку владение *открытым ключом* обеспечивает полный контроль над алгоритмом шифрования. Иначе говоря, против криптосистем с открытым ключом всегда можно организовать атаку на основе подобранного открытого текста. Итак, любую атаку на криптосистему с открытым ключом, в которой не используется блок расшифровки, можно назвать атакой CPA. Очевидно, что любая криптосистема с открытым ключом должна быть устойчивой к атакам на основе подобранного открытого текста, иначе от нее мало пользы.
- Как правило, криптосистемы с открытым ключом имеют стройную алгебраическую структуру, обладающую свойствами замкнутости, ассоциативности, гомоморфизма и т.п. (см. главу 5). Атакующий может использовать эти свойства и взломать зашифрованный текст с помощью хитроумных вычислений. Если атакующий имеет доступ к блоку расшифровки, его вычисления могут дать представление об исходном тексте и даже взломать всю криптосистему. Следовательно, криптосистемы с открытым ключом особенно уязвимы для атак CCA и CCA2. В главе показано, что все учебные криптосистемы с открытым ключом уязвимы для этих атак. Исходя из этого можно сформулировать следующее правило, основанное на свойстве 8.2.2: владелец открытого ключа никому не должен предоставлять услуги расшифровки. Это условие должно выполняться во всех криптосистемах, описанных в главе. В главе 14 будут рассмотрены более строгие криптосистемы с открытым ключом, не требующие от пользователей особой осторожности.
- На первый взгляд, атака CCA слишком ограничивает возможности атакующего. В приложениях пользователь, подвергающийся атаке (т.е. пользователь, к которому обратились за расшифровкой сообщения), на самом деле не знает об этом. Следовательно, пользователю не известно, когда он должен прекратить расшифровку. Как правило, предполагается, что пользователь слишком наивен и не предполагает о существовании атакующего, а значит, должен предоставлять услуги по расшифровке сообщений *постоянно*. С другой стороны, любая криптосистема с открытым ключом должна быть устойчивой к атакам CPA, поскольку атакующий может сам зашифровывать подобранные открытые сообщения. По этой причине, в основном, мы будем рассматривать средства защиты от атаки CCA2.

8.7 Задача RSA

Средства защиты криптосистемы RSA от атак CPA основаны на сложности вычисления корня e-й степени шифрованного текста c по составному целочисленному модулю n. Это — так называемая **задача RSA** (RSA problem).

Определение 8.4 (Задача RSA).
ИСХОДНЫЕ ДАННЫЕ:

$N = pq$, где p и q — простые числа;
e: целое число, удовлетворяющее условию $\gcd(e, (p-1)(q-1)) = 1$.
$c \in \mathbb{Z}_N^*$.

РЕЗУЛЬТАТ:

Единственное целое число $m \in \mathbb{Z}_N^*$, удовлетворяющее условию $m^e \equiv$ $\equiv c (\mathrm{mod}\, N)$.

Как и во всех других задачах, обеспечивающих стойкость криптосистем с открытым ключом, сложность задачи RSA зависит от сложности поиска правильных параметров.

Предположение 8.3 (Условия неразрешимости задачи RSA). *Алгоритмом решения задачи RSA называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[m \leftarrow \mathcal{A}\left(N, e, m^e (\mathrm{mod}\, N)\right)\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.4.

Пусть $\mathcal{IG}$ — генератор вариантов задачи RSA, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — 1) $2k$-битовый модуль $N = pq$, где p и q — два разных равномерно распределенных случайных простых числа, состоящих из k бит, и 2) $e \in \mathbb{Z}^*_{(p-1)(q-1)}$.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости задачи RSA, если для вариантов $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по отношению ко всем достаточно большим числам k.

Аналогично замечанию 8.1.3 (в разделе 8.4) можно утверждать, что выполнение условий неразрешимости задачи RSA означает существование однонаправленной хэш-функции. Как и в замечании 8.1.4, из этого следует, что она является функцией с секретом: существует эффективная процедура, обратная разложению модуля на простые множители.

Следует отметить, что вероятностное пространство в этом условии включает в себя пространство вариантов, пространство исходных сообщений и пространство случайных операций рандомизированного алгоритма решения задачи RSA.

Кроме того, в условии неразрешимости задачи RSA на вход предполагаемого алгоритма поступает показатель степени e, используемый при шифровании. Это ясно очерчивает цель атаки: решить задачу RSA при заданном показателе степени e. Существует альтернативная постановка задачи RSA, которая называется **сильной задачей RSA** (strong RSA problem) [85]. Ее цель — найти некоторый нечетный показатель степени $e > 1$ и решить задачу RSA для этого показателя. Очевидно, что решить сильную задачу RSA легче, чем обычную задачу RSA, в которой показатель степени e фиксирован. Считается, что сильная задача RSA является трудноразрешимой. **Условие трудной разрешимости сильной задачи RSA** легло в основу некоторых алгоритмов шифрования и криптографических протоколов.

Очевидно, что если для параметров открытого ключа (N, e) выполняется условие $m < N^{1/e}$, то шифрование сообщения $c = m^e (\mathrm{mod}\, N)$ не использует операции приведения по модулю, и, следовательно, значение m можно эффективно вычислить, извлекая корень e-й степени из целых чисел. Это одна из причин, по которой не следует выбирать вариант $e = 3$. В противном случае, если сообщение m зашифровано с помощью трех разных модулей: $c_i = m^3 (\mathrm{mod}\, N_i)$, где $i = 1, 2, 3$, то, поскольку модули являются попарно взаимно простыми числами, можно применить китайскую теорему об остатках (алгоритм 6.1) и вычислить значение $C = m^3 (\mathrm{mod}\, N_1 N_2 N_3)$. Теперь, поскольку $m < (N_1 N_2 N_3)^{1/3}$, возведение в степень выполняется точно так же, как и в пространстве целых чисел. Итак, расшифровка сообщения C сводится к извлечению кубического корня из целых чисел и может быть эффективно реализована (см. подсказку в упражнению 8.8).

Куперсмит (Coopersmith) [82] распространил этот тривиальный случай на более сложный: для $m' = m + t$, где число m известно, а число t — нет, но при этом выполняется неравенство $t < N^{1/e}$, значение t можно эффективно вычислить по заданному числу $c = m'^e (\mathrm{mod}\, N)$. Поскольку в приложениях часто встречаются ситуации, в которых часть исходного текста является известной (см. главу 15), считается, что при шифровании с помощью алгоритма RSA не следует применять очень маленькие показатели степени e. Как правило, в качестве показателя степени при шифровании используется величина $e = 2^{16} + 1 = 65537$, которая является простым числом. Этот показатель гарантирует достаточную эффективность шифрования и предотвращает атаки, использующие малые показатели степени.

Если показатель степени d, использующийся при расшифровке, невелик, алгоритм RSA устойчив к атакам на основе подобранного открытого текста. Винер (Wiener) изобрел метод, основанный на представлении числа e/N в виде непрерывной дроби. Этот алгоритм позволяет найти число d, если $d < N^{1/4}$ [298]. В дальнейшем этот результат был улучшен для чисел $d < N^{0,292}$ [50].

8.8 Разложение целых чисел на простые множители

Разрешимость задачи RSA зависит от сложности **разложения целого числа на простые множители** (integer factorization problem — IF-problem).

Определение 8.5 (Разложение целого числа на простые множители).
ИСХОДНЫЕ ДАННЫЕ:

N: нечетное составное целое число, имеющее по меньшей мере два разных простых множителя.

РЕЗУЛЬТАТ:

Простое число p, удовлетворяющее условию $p \mid N$.

Предположение 8.4 (Условие неразрешимости задачи о разложении целого числа на простые множители). *Алгоритмом решения задачи о разложении целого числа на простые множители называется вероятностный полиномиальный алгоритм $\mathcal{A}$ с преимуществом $\varepsilon > 0$:*

$$\varepsilon = \mathrm{Prob}\left[\mathcal{A}(N) \text{ является делителем числа } N \text{ и } 1 < \mathcal{A}(N) < N\right],$$

где входные данные алгоритма $\mathcal{A}$ указаны в определении 8.5.

Пусть $\mathcal{IG}$ — генератор целых чисел, получающий на вход число 1^k, время работы которого является полиномом от параметра k, а результатом — $2k$-битовый модуль $N = pq$, где p и q — два равномерно распределенных случайных простых нечетных числа, состоящих из k бит.

Будем говорить, что генератор $\mathcal{IG}$ удовлетворяет условиям неразрешимости задачи о разложении целого числа на простые множители, если для варианта $\mathcal{IG}(1^k)$ не существует алгоритма решения с преимуществом $\varepsilon > 0$, которое не является пренебрежимо малым по отношению ко всем достаточно большим числам k.

Очевидно, что алгоритм решения задачи о разложении целого числа на простые множители одновременно является алгоритмом решения задачи RSA, поскольку Алиса расшифровывает текст, зашифрованный с помощью алгоритма RSA, вычисляя значение $d \equiv e^{-1}(\mathrm{mod}(p-1)(q-1))$, т.е. используя информацию о факторизации числа N. Как и в задачах дискретного логарифмирования и Диффи–Хеллмана, обратное утверждение является недоказанным: выполняются ли условия неразрешимости задачи о разложении целого числа на множители, если условия неразрешимости задачи RSA не выполняются?

Как и в задаче Диффи–Хеллмана, слабым местом задачи о разложении целых чисел на простые множители являются гладкие простые числа N. Один из таких вариантов был продемонстрирован Поллардом, предложившим алгоритм эффективной факторизации под названием **$(\mathbf{p-1})$-алгоритм Полларда** [237]. В его

основе лежат следующие рассуждения. Пусть p — простой множитель числа N, а наибольший простой множитель числа $p-1$ ограничен величиной $B = \mathrm{Poly}(k)$, где $k = |N|$, а $\mathrm{Poly}(k)$ — полином, зависящий от аргумента k. (Число B называется **границей гладкости** числа $p-1$.) Сконструируем число

$$A = \prod_{\text{простые числа } r < B} r^{\lfloor \log N / \log r \rfloor}.$$

Условие $p - 1 \mid A$ выполняется автоматически, поэтому из малой теоремы Ферма (теорема 6.10) следует, что $a^A \equiv 1 (\mathrm{mod}\, p)$ для любого целого числа a, удовлетворяющего условию $\gcd(a, p) = 1$. Если существует простой множитель q числа N, не равный множителю p и такой что $a \not\equiv 1 (\mathrm{mod}\, q)$, то существует целое число ℓ, не кратное числу q, такое что $a^A - 1 (\mathrm{mod}\, N) = \ell p$. Следовательно, число $\gcd(a^A - 1 (\mathrm{mod}\, N), N)$ должно быть собственным простым множителем числа N. Если $N = pq$, это число должно равняться числу p. Осталось показать, что размер числа A полиномиально зависит от числа k, а значит, число $a^A (\mathrm{mod}\, N)$ можно найти за полиномиальное время.

По теореме о простом числе [170] существует не более чем $B/\log B$ простых чисел, которые меньше, чем число B. Отсюда следует, что

$$A < B^{\lfloor \log N \rfloor \frac{B}{\log B}} < B^{k \frac{B}{\log B}},$$

т.е.

$$|A| < kB \log 2 < k\,\mathrm{Poly}(k).$$

Очевидно, что правая часть этого неравенства представляет собой полином, зависящий от аргумента k. Следовательно, для вычисления значения $a^A (\mathrm{mod}\, N)$ можно выполнить многократное умножение по модулю N (алгоритм 4.3), причем количество операций умножения полиномиально зависит от аргумента k. Отметим, что число A не обязательно конструировать явно. Число $a^A (\mathrm{mod}\, N)$ можно найти путем вычисления величины $a^{r^{\lfloor \log N)/ \log r)\rfloor}} (\mathrm{mod}\, N)$ для всех простых чисел $r < B$.

Очень легко построить модуль $N = pq$ так, чтобы оценка гладкости чисел $p-1$ и $q-1$ была небольшой и неполиномиально зависела от $|N|$. В этом случае алгоритм RSA является стойким по отношению к данному методу факторизации. Можно начать с поиска больших простых чисел p' и q', таких что $p = 2p' + 1$ и $q = 2q' + 1$ также являются простыми. Такие простые числа называются **безопасными простыми числами** (safe prime), а модули алгоритма RSA, имеющие два безопасных простых множителя, называются **безопасными простыми модулями алгоритма RSA** (safe-prime RSA modulus). В настоящее время исследователи ведут дебаты о необходимости применения безопасных простых модулей в криптосистемах RSA. Противники их применения (см. работу [273]) утверждают, что модули алгоритма RSA должны быть как можно более случайными, и для

случайно выбранного простого числа p вероятность того, что число $p-1$ имеет большой простой множитель, является весьма большой. Однако корректность функционирования многих криптографических протоколов, основанных на сложности разложения целых чисел на множители, обеспечивается именно выбором безопасных простых модулей алгоритма RSA.

Хорошо известно, что частичная информация о простом множителе N позволяет разработать эффективный алгоритм факторизации числа N. Например, если $N = pq$, где p и q — простые числа, имеющие приблизительно одинаковый размер, знание половины битов числа p позволяет разложить число N за время, полиномиально зависящее от его размера [82].

При отсутствии априорной информации о простых множителях входного составного числа наилучшим алгоритмом факторизации является метод решета числового поля (number field sieve — NFS), временна́я сложность которого определяется по формуле (4.6.1). Таким образом, как и при выборе параметра безопасности задачи дискретного логарифмирования в конечном поле, для обеспечения надежности в качестве нижней границы размера модуля алгоритма RSA широко используется число 1024.

Эффективность метода решета числового поля, реализованного с помощью большого количества параллельных компьютеров, была продемонстрирована в начале 2000 года: сеть, состоящая из 9 000 рабочих станций, разложила на множители модуль RSA, состоящий из 512 бит (RSA-512 Challenge), после четырех месяцев работы [70] параллельного алгоритма.

Исследования в области целочисленной факторизации ведутся очень интенсивно. Обзор методов решения задачи RSA дан в работе Бонэ (Boneh) [48]. Достижения и перспективы этой области исследований читатели могут найти в главе 3 книги [198].

8.9 Уязвимость учебного алгоритма RSA

В разделе 8.1 указано, что рассматриваемый вариант алгоритма является учебным и описан в большинстве учебников по криптографии. Перейдем к изучению свойств, обусловливающих стойкость (или нестойкость) учебного алгоритма RSA.

При случайном ключе и случайном сообщении, упомянутых в определении 8.4 и предположении 8.3, утверждение о существовании эффективной атаки на основе подобранного открытого текста должно быть ложным.

Теорема 8.1. *Криптосистема RSA устойчива к атаке на основе подобранного открытого текста по принципу "все или ничего" тогда и только тогда, когда выполняются условия неразрешимости задачи RSA.* □

Принцип “все или ничего” объясняется при описании свойства 8.2.1. А применение атаки на основе подобранного открытого текста означает, что атакующий остается пассивным, как и предусмотрено свойством 8.2.2.

Однако такая стойкость имеет не очень высокую практическую ценность.

Во-первых, рассмотрим принцип “все или ничего”. Слово “все” означает, что при расшифровке восстанавливается весь блок исходного сообщения: размеры сообщения и модуля совпадают. В приложениях это условие выполняется не всегда. В реальных системах исходный текст, как правило, содержит определенную часть открытой информации, известной атакующему. Учебный вариант алгоритма RSA не скрывает эту информацию. Например, если известно, что зашифрованный текст представляет собой число, не превосходящее 1 000 000 (например, секретное предложение цены или размер оклада), то, имея зашифрованное сообщение, атакующий может восстановить исходный текст менее чем за 1 000 000 попыток.

Как правило, исходный текст $m < N$ с ненулевой вероятностью можно восстановить, использовав только $\sqrt{m}$ попыток, если в распоряжении криптоаналитика есть память, имеющая размер $\sqrt{m}$. Этот факт установлен Бонэ, Жё (Joux) и Нгуеном (Nguen) [52]. Они исходили из того, что факторизация небольшого исходного сообщения не является трудноразрешимой задачей, и использовали мультипликативное свойство функции RSA:

$$(m_1 \times m_2)^e \equiv m_1^e \times m_2^e \equiv c_1 \times c_2 (\mathrm{mod}\, N). \tag{8.9.1}$$

Иначе говоря, факторизация исходного сообщения означает факторизацию соответствующего зашифрованного текста. Как правило, текст, зашифрованный с помощью алгоритма RSA, трудно разложить на простые множители из-за перемешивающего свойства функции шифрования, которое практически всегда приводит к тому, что размер зашифрованного текста равен размеру модуля. Однако из мультипликативного свойства следует, что если исходный текст поддается факторизации, то и зашифрованный текст — тоже. Это позволяет организовать атаку “встреча посередине” (“meet-in-the-middle” attack). Рассмотрим следующий пример.

Пример 8.3. Пусть $c = m^e (\mathrm{mod}\, N)$, причем Злоумышленнику известно число $m < 2^\ell$. Существует ненулевая вероятность того, что число m является составным и удовлетворяет условию

$$m = m_1 \cdot m_2, \text{ где } m_1, m_2 < 2^{\frac{\ell}{2}}. \tag{8.9.2}$$

Из мультипликативного свойства функции RSA следует, что

$$c = m_1^e \cdot m_2^e (\mathrm{mod}\, N). \tag{8.9.3}$$

Злоумышленник может создать упорядоченную базу данных

$$\left\{1^e, 2^2, 3^e, \ldots, \left(2^{\frac{\ell}{2}}\right)^e\right\} (\mathrm{mod}\, N)$$

и искать в ней число $c/i^e (\mathrm{mod}\, N)$, где $i = 1, 2, \ldots, 2^{\frac{\ell}{2}}$. Согласно формулам (8.9.2) и (8.9.3) поиск сводится к проверке условия

$$c/i^e \equiv j^e (\mathrm{mod}\, N)$$

и завершается за $2^{\frac{\ell}{2}}$ шагов, при выполнении которых вычисляется величина $i^e (\mathrm{mod}\, N)$. Поскольку Злоумышленнику известны числа i и j, он восстанавливает число $m = i \cdot j$.

Оценим затраты Злоумышленника. Размер базы данных равен $2^{\frac{\ell}{2}} \log N$ бит. Временны́е затраты складываются из следующих компонентов: сложность создания базы данных имеет порядок $O_B(2^{\frac{\ell}{2}} \log^3 N)$, сортировка $- O_B(\frac{\ell}{2} 2^{\frac{\ell}{2}})$, а поиск числа $j^e (\mathrm{mod}\, N)$ в упорядоченной базе $- O_B(2^{\frac{\ell}{2}} (\frac{\ell}{2} + \log^3 N))$. В последней оценке учитывается время, затраченное на возведение в степень по модулю и бинарный поиск (с помощью алгоритма 4.4). Итак, полная оценка временно́й сложности взлома имеет порядок $O_B(2^{\frac{\ell}{2}+1} (\frac{\ell}{2} + \log^3 N))$. Если Злоумышленнику доступна память, имеющая размер $2^{\frac{\ell}{2}} \log N$ бит, то временна́я сложность взлома намного меньше 2^ℓ. Эта атака по временно́й сложности сравнима с атакой по методу квадратного корня. □

Если размер исходного сообщения колеблется от 40 до 64 бит, то вероятность того, что исходный текст будет разложен на два одинаковых целых числа, изменяется от 18 до 50% [52].

Пример 8.4 (Реальный пример атаки 8.3). Представим себе сценарий, в котором ключ алгоритма DES, состоящий из 56 бит, зашифрован с помощью 1024-битового ключа учебного варианта RSA. Если ключ алгоритма DES случаен, его можно восстановить с ненулевой вероятностью, равной вероятности разложить ключ DES на два целых множителя длиной по 28 бит. Для этого понадобится память, имеющая размер 2^{38} бит (т.е. 32 гигабайт) и 2^{29} операций возведения в степень по модулю. Этим условиям вполне соответствует хороший персональный компьютер. В то же время непосредственный перебор ключей алгоритма DES требует выполнения 2^{56} операций возведения в степень по модулю. Это требование уже не под силу даже специализированному устройству. □

Теперь понятно, почему не следует применять учебный вариант алгоритма RSA для шифрования коротких ключей и паролей, длина которых не превышает 2^{64} бит. Что же произойдет, если в конкретном приложении нам понадобится

зашифровать с помощью алгоритма RSA небольшие числа, даже если размер сообщения равен одному биту? В этой ситуации следует применять методы шифрования, описанные в главе 15.

Следующий пример еще раз демонстрирует уязвимость учебного алгоритма RSA для атаки на основе подобранного открытого текста: против активной атаки этот алгоритм еще более беззащитен.

Пример 8.5. Допустим, что Злоумышленник частично контролирует блок шифрования, принадлежащий Алисе. Это условие вполне "разумно": если результат расшифровки зашифрованного текста, посланный Злоумышленником, окажется бессмысленным набором случайных цифр, Алиса должна вернуть Злоумышленнику исходный текст. "Разумность" этого условия объясняется двумя причинами.

1. "Случайный ответ на случайный запрос" — это вполне стандартный режим во многих криптографических протоколах, и, следовательно, пользователь должен выполнить определенную инструкцию "оклик–отзыв". Действительно, довольно *часто* криптографические протоколы допускают частичный контроль блока расшифровки со стороны пользователей. Например, протокол аутентификации с открытым ключом Нидхема–Шредера (протокол 2.5) функционирует именно так: Алиса обязана выполнить расшифровку зашифрованного текста, полученного от Боба.
2. В любом случае хочется надеяться, что расшифрованный текст, имеющий вид случайного набора цифр, не позволит атакующему извлечь какую-либо полезную информацию.

Предположим теперь, что Злоумышленник желает знать исходный текст, лежащий в основе зашифрованного текста $c \equiv m^e (\mathrm{mod}\, N)$, перехваченного им в ходе предыдущего сеанса связи между Алисой и кем-либо еще (но не с ним!). Злоумышленник извлекает случайное число $r \in_U \mathbb{Z}_N^*$, вычисляет зашифрованный текст $c' = r^e c (\mathrm{mod}\, N)$ и посылает его Алисе. Результат расшифровки, выполненной Алисой, равен

$$c'^d \equiv rm (\mathrm{mod}\, N).$$

Это число может показаться Алисе совершенно случайным, поскольку умножение на число r является **перестановкой** над группой $\mathbb{Z}_N^*$. Итак, Алиса возвращает результат расшифровки rm Злоумышленнику. Увы! Злоумышленнику известно число r, и он может вычислить число m с помощью деления по модулю N. □

Примеры 8.3–8.5 демонстрируют, что учебный алгоритм RSA слишком слаб для реальных приложений. Необходимо систематически исправлять его недостатки. Эта работа выполняется за два этапа.

- В главе 14 вводятся понятия повышенной стойкости схем шифрования с открытым ключом, пригодные для реальных приложений.

- В главе 15 изучается реальная версия алгоритма шифрования RSA, который является стандартным. В ней приводится формальное доказательство стойкости алгоритма RSA на основе понятия сильной стойкости.

8.10 Криптосистема Рабина (учебный вариант)

Криптосистема, разработанная Рабином (Rabin), основана на сложности вычисления квадратного корня по модулю составного числа [240]. Работа Рабина носила теоретический характер. В ней впервые приводилось доказательство стойкости криптосистем с открытым ключом: стойкость криптосистемы Рабина эквивалентна неразрешимости задачи о разложении целых чисел на множители. (Напомним, что эквивалентность разрешимости задачи RSA и разрешимости задачи о разложении целых чисел на множители еще не доказана.) Алгоритм шифрования в криптосистеме Рабина чрезвычайно эффективен и пригоден для многих практических приложений, например, для шифрования с помощью портативных устройств.

Криптосистема Рабина описывается алгоритмом 8.2.

Алгоритм 8.2. Криптосистема Рабина

Ключ

Для создания ключа Алиса должна выполнить следующие действия.

1. Выбрать два случайных простых числа p и q, удовлетворяющих условию $|p| \approx |q|$.
   (* Этот этап совпадает с вычислением модулей алгоритма RSA в алгоритме 8.1. *)
2. Найти число $N = pq$.
3. Извлечь случайное целое число $b \in_U \mathbb{Z}_N^*$.
4. Использовать пару (N, b) в качестве параметров открытого ключа и запомнить пару (p, q) в качестве параметров закрытого ключа.

Шифрование

Для того чтобы послать Алисе секретное сообщение $m \in \mathbb{Z}_N^*$, Боб создает зашифрованный текст c:

$$c \leftarrow m(m + b) (\mathrm{mod}\, N).$$

Расшифровка

Для того чтобы расшифровать зашифрованный текст c, Алиса решает квадратное уравнение

$$m^2 + bm - c \equiv 0 (\mathrm{mod}\, N),$$

где $m < N$.

Покажем, что алгоритм 8.2 действительно описывает криптосистему, т.е. процедура расшифровки, выполняемая Алисой, действительно восстанавливает исходный текст, зашифрованный Бобом.

Из элементарной математики известно, что общее решение квадратного уравнения имеет вид:

$$m \equiv \frac{-b + \sqrt{\Delta_c}}{2} \pmod{N}, \tag{8.10.1}$$

где

$$\Delta_c \stackrel{\text{def}}{=} b^2 + 4c \pmod{N}. \tag{8.10.2}$$

Поскольку число c зависит от элемента $m \in \mathbb{Z}_N^*$, квадратное уравнение

$$m^2 + bm - c \equiv 0 \pmod{N}$$

имеет решения в группе $\mathbb{Z}_N^*$. Одним из этих решений является число m, посланное Бобом. Отсюда следует, что число Δ_c должно быть квадратичным вычетом по модулю N, т.е. элементом группы QR_N.

Вычисления, связанные с расшифровкой, содержат операцию извлечения квадратного корня по модулю N. В разделе 6.6.2 показано, что вычислительная сложность этой задачи эквивалентна факторизации числа N (следствие 6.3). Следовательно, Алиса является единственным пользователем, который может вычислить формулу (8.10.1), поскольку только ей известно разложение числа N на простые множители. Алиса может найти число $\sqrt{\Delta_c}$, используя алгоритм 6.5. В разделе 6.6.2 показано, что для каждого зашифрованного текста c, посланного Бобом, существует четыре разных значения $\sqrt{\Delta_c}$, и, следовательно, существует четыре разных результата шифрования. Как правило, реальное исходное сообщение содержит *избыточную информацию* (redundant information), позволяющую Алисе отличать правильные тексты от неправильных. Смысл понятия "избыточная информация" и способ ее внедрения в текст изложены в разделе 10.4.3.

Отметим, что, если число N является **целым числом Блюма** (Blum integer), т.е. $N = pq$, где $p \equiv q \equiv 3 \pmod{4}$, вычисление квадратных корней по модулю N упрощается и сводится к вычислению квадратных корней по модулю p и q с помощью алгоритма 6.3 для $p \equiv 3, 7 \pmod{8}$ и применения китайской теоремы об остатках. Следовательно, на практике открытые модули, применяемые в криптосистеме Рабина, должны быть целыми числами Блюма.

В алгоритме шифрования Рабина используется только одно умножение и одно сложение. Следовательно, этот алгоритм работает быстрее, чем алгоритм шифрования RSA.

Пример 8.6. Допустим, что Алиса выбирает числа $N = 11 \times 19 = 209$ и $b = = 183$. Она объявляет пару $(N, b) = (209, 183)$ параметрами открытого ключа криптосистемы Рабина.

Предположим, что Боб зашифровал исходное сообщение $m = 31$ с помощью алгоритма шифрования Рабина.

$$c = 31 \times (31 + 183) \equiv 155 (\mathrm{mod}\, 209).$$

Результирующий зашифрованный текст равен 155.

Для расшифровки зашифрованного текста, равного 155, Алиса вычисляет величину Δ_c по формуле (8.10.2).

$$\Delta_c = b^2 + 4c = 183^2 + 4 \times 155 \equiv 42 (\mathrm{mod}\, 209).$$

Теперь, применяя алгоритм 6.5, Алиса определяет четыре квадратных корня числа 42 по модулю 209, т.е. числа 135, 173, 36, 74. В заключение она может применить формулу (8.10.1) и получить четыре результата расшифровки: 185, 204, 31 и 50. В реальной криптосистеме Рабина исходный текст должен содержать дополнительную информацию, позволяющую распознать среди полученных ответов искомый результат. □

8.11 Уязвимость учебной криптосистемы Рабина

Существует чрезвычайно эффективная активная атака против криптосистемы Рабина. В ее основе лежит следующая теорема.

Теорема 8.2.

1. *Криптосистема Рабина является доказуемо стойкой к атаке на основе подобранного открытого зашифрованного текста в рамках подхода "все или ничего", если и только если задача о разложении целого числа на простые множители является трудноразрешимой.*
2. *Криптосистема Рабина является абсолютно беззащитной перед атакой на основе подобранного зашифрованного текста.*

Доказательство. I. Поскольку процедура расшифровки в криптосистеме Рабина использует разложение модулей алгоритма RSA на простые множители, стойкость криптосистемы Рабина означает невозможность факторизации этих модулей. Следовательно, достаточно показать обратное утверждение: неразрешимость задачи о разложении целого числа на простые множители означает стойкость криптосистемы Рабина.

Допустим, что существует оракул $\mathcal{O}$, взламывающий криптосистему Рабина с ненулевым преимуществом ε, т.е.

$$\mathrm{Prob}\left[\mathcal{O}(c, N) = \frac{-b + \sqrt{\Delta_c}}{2} (\mathrm{mod}\, N) \mid c \in_U Z_N^*\right] \geqslant \varepsilon.$$

Выберем случайное сообщение m, определим число $c = m(m+b) (\mathrm{mod}\, N)$ и вызовем оракула $\mathcal{O}(c, N)$, который возвращает число $m' \equiv \frac{-b+\sqrt{\Delta_c}'}{2} (\mathrm{mod}\, N)$ с преимуществом ε. Здесь $\sqrt{\Delta_c}'$ обозначает один из четырех квадратных корней числа Δ_c. Из теоремы 6.17 (раздел 6.6.2) следует, что с вероятностью $1/2$ выполняется неравенство

$$m' + \frac{b}{2} \equiv \frac{\sqrt{\Delta_c}'}{2} \not\equiv \pm \frac{\sqrt{\Delta_c}}{2} \equiv \pm \left(m + \frac{b}{2} \right) (\mathrm{mod}\, N) .$$

Однако, поскольку

$$\left(m' + \frac{b}{2} \right)^2 \equiv \frac{\Delta_c}{4} \equiv \left[\pm \left(m' + \frac{b}{2} \right) \right]^2 (\mathrm{mod}\, N) ,$$

из теоремы 6.17 следует, что

$$\gcd \left(m' + \frac{b}{2} \pm \left(m + \frac{b}{2} \right), N \right) = p \text{ или } q. \tag{8.11.1}$$

Иначе говоря, число N можно разложить на множители с ненулевым преимуществом $\varepsilon/2$. Это противоречит предположению о невозможности факторизации модулей алгоритма RSA при условиях неразрешимости задачи о разложении целых чисел на простые множители. Итак, утверждение I доказано.

Утверждение II является тривиальным, если атакующий может получить доступ к блоку расшифровки сообщений: в этом случае блок расшифровки играет роль оракула, использованного в доказательстве утверждения I! Поскольку атакующий генерирует зашифрованный текст для его последующей расшифровки оракулом, такая атака является атакой на основе подобранного зашифрованного текста. □

Из теоремы 8.2 следуют два вывода. Во-первых, криптосистема Рабина является доказуемо стойкой в рамках подхода “все или ничего”, описанного свойством 8.2.1, и зависит от сложности разложения целых чисел на простые множители (если текст не содержит никакой *априорной* информации). Это значительный результат, поскольку он относится к стойкости учебной схемы шифрования Рабина. Если задача о разложении целых чисел на простые множители является трудноразрешимой, то предполагаемый оракул $\mathcal{O}$ в доказательстве утверждения I не может существовать. Однако следует уделить особое внимание модификации принципа “все или ничего” в отношении стойкости криптосистемы по отношению к атакам на основе подобранного открытого текста. Здесь слово “все” означает, что *в общем случае* восстанавливается весь блок исходного текста, т.е. размер сообщения совпадает с размером модуля. Очевидно, что, поскольку шифрование

Рабина является детерминированным алгоритмом, в некоторых случаях расшифровка сообщений не обязательно так же сложна, как и разложение целых чисел на простые множители. Мы еще вернемся к этому утверждению, когда будем рассматривать атаку "встреча посередине" на схему Рабина в конце этого раздела.

Во-вторых, очевидно, что в криптосистеме Рабина *никто* и *никогда* не должен играть роль оракула. Атака на основе подобранного зашифрованного текста уничтожает криптосистему Рабина: в результате такой атаки восстанавливается не только *отдельная* информация об исходном тексте (как в случае атаки CCA2 против криптосистемы RSA в примере 8.5), но и раскрывается секретный ключ. Следовательно, атакующий получает возможность читать *все* секретные сообщения, зашифрованные с помощью открытого ключа.

Пример 8.7. В примере 8.6, посвященном криптосистеме Рабина, были рассмотрены параметры открытого ключа $(N, b) = (209, 183)$ и четыре результата расшифровки сообщения 31: числа $185, 204, 31$ и 50.

Если эти числа станут известны постороннему лицу, не владеющему открытым ключом, например, в результате атаки CCA, он может использовать их для факторизации числа 209. Применяя формулу (8.11.1), получаем, что

$$\gcd(204 - 185, 209) = 19$$

или

$$\gcd((31 + 183/2) + (50 + 183/2), 209) = \gcd(264, 209) = 11.$$ □

Хотя мы предупреждали, что владелец открытого ключа в схеме шифрования Рабина не должен предоставлять услуги расшифровки, в реальных приложениях не стоит полагаться на высокую бдительность пользователей. Следовательно, учебный вариант схемы шифрования Рабина не пригоден для практического применения. Реальные варианты криптосистем Рабина и RSA будут описаны в главе 15. Там же приводится формальное доказательство стойкости практических схем шифрования.

Следует отметить, что в криптосистемах Рабина и RSA используются одни и те же модули. Следовательно, на криптосистему Рабина распространяются меры предосторожности, касающиеся выбора модулей в криптосистеме RSA.

В заключение отметим, что атака "встреча посередине" позволяет взломать еще один вариант схемы шифрования Рабина.

Шифрование. $c = m^2 (\mathrm{mod}\, N)$.

Расшифровка. Вычисление квадратного корня по модулю N.

Как и в учебной криптосистеме RSA, легкость факторизации небольших исходных сообщений (см. раздел 8.9) в криптосистеме Рабина обеспечивает успех атаки на основе перехвата, описанной в примере 8.3.

8.12 Криптосистема Эль-Гамаля (учебный вариант)

Эль-Гамаль разработал весьма остроумную криптосистему с открытым ключом [102], использующую однонаправленную функцию Диффи–Хеллмана с секретом. Работа Эль-Гамаля вызвала большой теоретический и практический интерес, который сохраняется до сих пор. В дальнейшей части книги мы рассмотрим два усовершенствованных варианта этой криптосистемы — схему шифрования Эль-Гамаля (глава 13) и криптосистему Эль-Гамаля с доказуемо высокой стойкостью (глава 15).

Одна из причин широкой популярности этой криптосистемы заключается в использовании задачи Диффи–Хеллмана, неразрешимость которой общепризнанна. Считается, что ее сложность эквивалентна сложности задачи о вычислении дискретного логарифма, которая, в свою очередь, является альтернативой задачи о разложении целого числа на простые множители, лежащей в основе криптосистем RSA и Рабина.

Криптосистема Эль-Гамаля описывается алгоритмом 8.3.

Покажем, что система, описанная алгоритмом 8.3, действительно является криптографической, т.е. в результате расшифровки Алиса восстанавливает тот самый исходный текст, который был послан Бобом.

Поскольку

$$c_1^x \equiv \left(g^k\right)^x \equiv (g^x)^k \equiv y^k \equiv \frac{c_2}{m} (\mathrm{mod}\, p),$$

формула (8.12.2) действительно позволяет восстановить исходный текст m.

Для деления, выполняемого в формуле (8.12.2), необходимо применить расширенный алгоритм Евклида (алгоритм 4.2), что в общем случае сопровождается бóльшими затратами, чем умножение. Однако Алиса может избежать деления, использовав следующие вычисления.

$$m \leftarrow c_2 c_1^{-x} (\mathrm{mod}\, p).$$

Легко проверить, что эта схема расшифровки вполне работоспособна, однако следует иметь в виду, что число $-x$ в этой формуле на самом деле означает число $p - 1 - x$.

Пример 8.8. В примере 8.1 показано, что число 3 является первообразным корнем по модулю 43. Допустим, Алиса выбирает в качестве своего закрытого ключа число 7. Затем она вычисляет свой открытый ключ:

$$37 \equiv 3^7 (\mathrm{mod}\, 43).$$

После этого Алиса оглашает параметры своего открытого ключа $(p, g, y) = (43, 3, 37)$.

Алгоритм 8.3. Криптосистема Эль-Гамаля

Ключ

Для создания ключа Алиса должна выполнить следующие действия.

1. Выбрать случайное простое число p.
2. Вычислить случайный мультипликативный порождающий элемент $g \in \mathbb{F}_p^*$.
3. Извлечь случайное целое число $x \in {}_U\mathbb{Z}_{p-1}$ и считать его своим закрытым ключом.
4. Вычислить открытый ключ
$$y \leftarrow g^x (\mathrm{mod}\, p).$$
5. Использовать тройку (p, g, y) в качестве параметров открытого ключа и запомнить число x в качестве закрытого ключа.

(* Аналогично протоколу обмена ключами Диффи–Хеллмана системные пользователи могут использовать общие открытые параметры (p, g). *)

Шифрование

Для того чтобы послать Алисе секретное сообщение $m < p$, Боб извлекает случайное целое число $k \in {}_U\mathbb{Z}_{p-1}$ и вычисляет зашифрованный текст (c_1, c_2):

$$\begin{cases} c_1 \leftarrow g^k (\mathrm{mod}\, p), \\ c_2 \leftarrow y^k m (\mathrm{mod}\, p). \end{cases} \tag{8.12.1}$$

Расшифровка

Для того чтобы расшифровать зашифрованный текст (c_1, c_2), Алиса вычисляет формулу

$$m \leftarrow \frac{c_2}{c_1^x} (\mathrm{mod}\, p). \tag{8.12.2}$$

Предположим, что Боб шифрует исходное сообщение $m = 14$. Он извлекает случайный показатель степени, равный 26, и вычисляет следующие значения.

$$c_2 = 15 \equiv 3^{26} (\mathrm{mod}\, 43), c_2 = 31 \equiv 36^{26} \times 14 (\mathrm{mod}\, 3).$$

В результате возникает зашифрованный текст $(15, 31)$.

Для того чтобы расшифровать сообщение $(15, 31)$, Алиса находит следующее число.

$$14 = 31/36 \equiv 31/15^7 (\mathrm{mod}\, 43).$$

Для деления необходимо применить алгоритм 4.2. Однако Алиса не делает этого и вычисляет значение 14 иначе.

$$14 = 31 \times 15^{42-7} \equiv 31 \times 6 (\mathrm{mod}\, 43).$$

□

8.13 Уязвимость учебной криптосистемы Эль-Гамаля

Алгоритм шифрования (8.12.1) является вероятностным: он использует случайное число $k \in_U \mathbb{Z}_{p-1}$. Допустим, что закрытый ключ Алисы x является числом, взаимно простым с числом $p-1$. Тогда по теореме 5.2.3 (раздел 5.2.3) открытый ключ Алисы $y \equiv g^x (\mathrm{mod}\, p)$, как и число g, является порождающим элементом группы $\mathbb{F}_p^*$. Следовательно, число $y^k (\mathrm{mod}\, p)$ пробегает всю группу $\mathbb{F}_p^*$, когда элемент k пробегает группу $\mathbb{Z}_{p-1}$. Поскольку умножение по модулю p является перестановкой над группой $\mathbb{F}_p^*$, для любого исходного сообщения $m \in \mathbb{F}_p^*$ число $c_2 \equiv y^k m (\mathrm{mod}\, p)$ пробегает всю группу $\mathbb{F}_p^*$, когда число k пробегает группу $\mathbb{Z}_{p-1}$ (теорема 6.6 из раздела 6.2.2). Итак, если $k \in_U \mathbb{Z}_{p-1}$, то $c_2 \in_U \mathbb{F}_p^*$. Это означает, что шифрование Эль-Гамаля *равномерно* распределяет исходное сообщение по всему пространству сообщений. Это — идеальное семантическое свойство алгоритма шифрования.

Однако не следует быть слишком большими оптимистами! Зашифрованный текст в схеме Эль-Гамаля — это не отдельный блок c_2, а пара (c_1, c_2), причем числа c_1 и c_2 статистически *связаны* между собой. Следовательно, как и все криптосистемы с открытым ключом, стойкость криптосистемы Эль-Гамаля зависит от условий неразрешимости задачи, лежащей в ее основе. Более того, как будет показано в разделе 8.13.1, для того, чтобы обеспечить идеальное семантическое свойство, исходное сообщение должно принадлежать группе $\langle g \rangle$. К сожалению, в реальных системах это условие, как правило, не выполняется.

Рассмотрим стойкость криптосистемы Эль-Гамаля по принципу "все или ничего".

Теорема 8.3. *Для исходного сообщения, равномерно распределенного по всему пространству исходных сообщений, криптосистема Эль-Гамаля является стойкой к атаке на основе подобранного открытого текста по принципу "все или ничего" тогда и только тогда, когда задача Диффи–Хеллмана является трудноразрешимой.*

Доказательство. ($\Rightarrow$) Необходимо показать, что, если криптосистема Эль-Гамаля является стойкой, выполняются условия неразрешимости задачи Диффи–Хеллмана.

Допустим противное — условия неразрешимости задачи Диффи–Хеллмана не выполняются. Тогда при заданном тексте $(c_1, c_2) \equiv (g^k, y^k m) (\mathrm{mod}\, p)$, зашифрованном с помощью открытого ключа $y \equiv g^x (\mathrm{mod}\, p)$, оракул задачи Диффи–Хеллмана вычислит по четверке (p, g, g^x, g^k) число $g^{xk} \equiv y^k (\mathrm{mod}\, p)$ с ненулевым преимуществом. Затем с тем же преимуществом вычисляется величина $m \leftarrow c_2 / y^k (\mathrm{mod}\, p)$. Это противоречит стойкости криптосистемы Эль-Гамаля.

($\Leftarrow$) Теперь необходимо показать, что при условиях неразрешимости задачи Диффи–Хеллмана не существует ни одного эффективного алгоритма, позволяющего с преимуществом, которое не является пренебрежимо малым, восстановить исходное сообщение, зашифрованное с помощью криптосистемы Эль-Гамаля.

Допустим противное — существует эффективный оракул $\mathcal{O}$, позволяющий взламывать криптосистему Эль-Гамаля, т.е. для любых параметров открытого ключа (p, g, y) и зашифрованного текста (c_1, c_2) оракул $\mathcal{O}$ с преимуществом δ, которое не является пренебрежимо малым, вычисляет значение

$$m \leftarrow \mathcal{O}(p, g, y, c_1, c_2),$$

удовлетворяющее условию

$$\frac{c_2}{m} \equiv g^{\left(\log_g y \log_g c_1\right)} (\mathrm{mod}\, p).$$

Затем для произвольного варианта задачи Диффи–Хеллмана (p, g, g_1, g_2) тройка (p, g, g_1) выбирается в качестве параметров открытого ключа, а пара (g_2, c_2) — в качестве зашифрованной пары, где $c_2 \in \mathbb{F}_p^*$. Тогда с преимуществом δ оракул $\mathcal{O}$ вычисляет значение

$$m \leftarrow \mathcal{O}(p, g, g_1, g_2, c_2),$$

которое удовлетворяет условию

$$\frac{c_2}{m} \equiv g^{\left(\log_g g_1 \log_g g_2\right)} (\mathrm{mod}\, p).$$

Это противоречит условиям неразрешимости задачи Диффи–Хеллмана. □

Поскольку стойкость криптосистемы Эль-Гамаля к атаке CPA эквивалентна неразрешимости задачи Диффи–Хеллмана, все утверждения, касающиеся параметров открытого ключа (раздел 8.4), относятся и к криптосистеме Эль-Гамаля. Как и в протоколе обмена ключами Диффи–Хеллмана, криптосистема Эль-Гамаля работает в подгруппе группы $\mathbb{F}_q$, порядок которой является большим простым числом, или в большой группе точек эллиптической кривой, определенной над конечным полем.

8.13.1 Атака “встреча посередине” и активная атака на учебную криптосистему Эль-Гамаля

Алгоритм 8.3, описывающий криптосистему Эль-Гамаля, является учебным, поскольку его схема шифрования очень слаба. Попробуем разобраться в причинах ее слабости.

Обычная схема шифрования Эль-Гамаля, применяемая в реальных приложениях, допускает частичную утечку информации даже при пассивных атаках. На практике в криптосистеме Эль-Гамаля для достижения большей эффективности часто используется элемент g, имеющий порядок $r = \operatorname{ord}_p(g) \ll p$. Если в этом случае сообщение m не принадлежит подгруппе $\langle g \rangle$, то атака “встреча посередине” на криптосистему Эль-Гамаля почти не отличается от подобной атаки на криптосистему RSA (см. пример 8.3). По зашифрованному тексту $(c_1, c_2) = (g^k, y^k m) (\mathrm{mod}\, p)$ Злоумышленник может найти число

$$c_2^r \equiv m^r (\mathrm{mod}\, p).$$

Таким образом, Злоумышленник преобразовал “вероятностную” схему шифрования в криптосистеме Эль-Гамаля в *детерминированную*! Более того, криптосистема Эль-Гамаля обладает таким же мультипликативным свойством, что и криптосистема RSA (см. раздел 8.9). Следовательно, при пересылке коротких сообщений, которые легко разложить на простые множители, Злоумышленник может организовать атаку “встреча посередине” точно так же, как это делается в криптосистеме RSA [52].

Итак, если исходное сообщение не принадлежит подгруппе, порожденной элементом g, криптосистема Эль-Гамаля становится детерминированной. Такие криптосистемы допускают частичную утечку информации, поскольку уязвимы для атаки путем перебора коротких исходных сообщений, например, секретных предложений цены или зарплат сотрудников.

Покажем теперь, что криптосистема Эль-Гамаля уязвима для активной атаки.

Пример 8.9. Допустим, что Злоумышленник имеет частичный доступ к блоку расшифровки Алисы в криптосистеме Эль-Гамаля. Как и в примере 8.5, разумно предположить, что в ответ на бессмысленное сообщение, полученное от Злоумышленника, Алиса должна вернуть ему результат расшифровки.

Предположим, что Злоумышленник владеет зашифрованным текстом $(c_1, c_2) \equiv \equiv (g^k, y^k m) (\mathrm{mod}\, p)$, перехваченным им во время предыдущего сеанса связи между Алисой и кем-либо еще (кроме Злоумышленника!). Если Злоумышленник желает восстановить исходный текст, он генерирует случайное число $r \in_U \mathbb{F}_p^*$, вычисляет $c_2' = rc_2 (\mathrm{mod}\, p)$ и посылает подобранный шифрованный текст (c_1, c_2') Алисе. После расшифровки Алиса восстанавливает число

$$rm (\mathrm{mod}\, p),$$

которое, с ее точки зрения, выглядит совершенно случайным, поскольку умножение на число $r < p$ является перестановкой над группой $\mathbb{F}_p^*$. Алиса возвращает Злоумышленнику число rm. Увы! Злоумышленнику известно число r, и он может восстановить число m, выполняя операцию деления по модулю p. □

8.14 Необходимость понятия повышенной стойкости для криптосистем с открытым ключом

Мы рассмотрели несколько учебных криптосистем с открытым ключом. Их можно считать непосредственными приложениями разных однонаправленных функций с секретом. (Понятие однонаправленной функции с секретом введено при описании свойства 8.1.) Настало время разобраться в причинах уязвимости этих криптосистем. Следует обсудить два аспекта уязвимости учебных криптосистем с открытым ключом.

Во-первых, как указано при описании свойства 8.2.1, в рамках этой главы мы рассматриваем очень слабое понятие стойкости: по принципу "все или ничего". В большинстве приложений криптосистем с открытым ключом такая стойкость совершенно неприемлема. Как правило, исходные сообщения содержат *априорную* информацию, известную атакующему. Например, если шифруется результат голосования, то атакующий априори знает, что ответом может быть либо слово "ДА", либо слово "НЕТ", либо фамилия кандидата. Таким образом, независимо от стойкости функции с секретом, атакующему достаточно применить метод перебора и подобрать исходный текст. В других приложениях атакующий получает дополнительное преимущество за счет знания априорной информации (пример такой атаки будет рассмотрен в разделе 14.3.2). Как правило, учебный алгоритм шифрования недостаточно хорошо скрывает подобную информацию. Следовательно, необходимо разработать криптосистемы с открытым ключом, скрывающие любую априорную информацию.

Во-вторых, как указывалось при описании свойства 8.2.2, в рамках этой главы рассматривается очень слабый вариант атак — пассивные атаки. Однако для каждой из рассмотренных учебных криптосистем с открытым ключом были продемонстрированы и активные атаки (примеры 8.5, 8.7 и 8.9). В ходе атаки CCA или CCA2 Злоумышленник может подготовить очень хитроумное сообщение и переслать его владельцу ключа для дальнейшей расшифровки. Как показано выше, учебные криптосистемы с открытым ключом весьма уязвимы для таких атак. Несмотря на меры предосторожности, которые можно предпринять в таких ситуациях, их явно недостаточно. Невозможно держать пользователей в состоянии

постоянной тревоги и запрещать им отвечать на какие бы то ни было запросы, подлежащие расшифровке.

Различными авторами были разработаны криптосистемы с открытым ключом повышенной стойкости. В главе 14 будут введены понятия, связанные с повышенной стойкостью, и показано, как достичь **формально доказуемой стойкости** (formally provable security). В главе 15 мы рассмотрим практичные криптосистемы с открытым ключом, стойкость которых можно доказать даже в рамках очень строгого подхода.

8.15 Комбинация асимметричной и симметричной криптографии

Криптография с открытым ключом весьма элегантно решает задачу распределения ключей. Однако, как правило, криптографические функции с открытым ключом действуют в очень крупных алгебраических структурах, т.е. связаны с выполнением весьма затратных алгебраических операций. С этой точки зрения симметричные криптографические функции гораздо эффективнее. Например, алгоритм AES работает в поле, состоящем из 256 элементов. Его основные операции, такие как умножение и вычисление обратной функции, можно выполнить с помощью очень эффективного табличного поиска (см. раздел 7.7.4). Кроме того, вычисления в криптосистемах с открытым ключом выполняются намного интенсивнее, чем в их симметричных аналогах.

В приложениях, особенно связанных с шифрованием больших объемов данных, стандартным решением стало использование **гибридных схем** (hybrid scheme). В таких схемах криптография с открытым ключом используется для шифрования так называемого **эфемерного ключа** (ephemeral key) для симметричной криптосистемы. Этот ключ распределяется между отправителем и адресатом, а затем с его помощью шифруется большой массив данных. Гибридные системы обладают лучшими свойствами обеих криптосистем: легкостью распределения ключей в криптосистемах с открытым ключом и эффективностью симметричных криптосистем.

Широкое распространение получила комбинация криптосистем с открытым и симметричным ключами — **цифровой конверт** (digital envelope). Эта схема является комбинацией криптосистемы RSA с симметричной криптосистемой, например, с алгоритмом DES, тройным алгоритмом DES или алгоритмом AES. Такая комбинация (RSA+DES или RSA+тройной DES) представляет собой схему **безопасного переходника** (secure sockets layer — SSL) [136]. Протокол SSL описан в главе 12. Он широко применяется в Web-браузерах, таких как Netscape и Internet Explorer, а также в Web-серверах. Инициатор протокола SSL (допустим, Алиса — как правило, пользователь сети Web) загружает параметры открытого ключа дру-

гой стороны (например, Боба — как правило, Web-сервера). Затем Алиса (точнее, ее Web-браузер) генерирует случайный сеансовый ключ, шифрует его ("запечатывает в конверт"), используя открытый ключ Боба, и посылает Бобу. После этого Боб (точнее, его Web-сервер) расшифровывает сообщение ("распечатывает конверт") и восстанавливает сеансовый ключ. Теперь обе стороны могут использовать этот ключ в симметричной схеме шифрования для дальнейшего обмена секретными сообщениями.

С точки зрения организации протокола гибридная схема очень проста. Однако существует два ограничения. Во-первых, эта схема использует сеансовый ключ, созданный одной из сторон (отправителем сообщения, или инициатором протокола), а другая сторона (получатель сообщения, или адресат протокола) должен целиком полагаться на компетентность и честность инициатора протокола. В некоторых ситуациях это нежелательно: например, в протоколе SSL, в рамках которого клиентом является отправитель, а точнее — его программное обеспечение, которое, как известно, представляет собой весьма слабый датчик случайных чисел.

Во-вторых, гибридные схемы шифрования инертны. В таких системах перехватчик, который может силой заставить получателя раскрыть свой закрытый ключ, получает возможность расшифровывать все сообщения. Эта особенность называется недостатком "заблаговременной секретности". Заблаговременная секретность означает, что перехватчик не в состоянии расшифровать исходное сообщение в будущем, используя зашифрованные сообщения, полученные в прошлом, ни с помощью криптоанализа, ни с помощью *принуждения*.

Эти ограничения можно преодолеть, если в качестве криптосистемы с открытым ключом использовать протокол обмена ключами Диффи–Хеллмана.

Рассмотрим сначала, как преодолевается первое ограничение. При запуске протокола обмена ключами Диффи–Хеллмана между Алисой и Бобом распределяемый секрет g^{ab} представляет собой случайное число, сформированное обеими сторонами: вклад Алисы — число a, вклад Боба — число b. Если число g порождает группу, имеющую простой порядок, и сообщения протокола удовлетворяют условиям $g^a \neq 1$ и $g^b \neq 1$ (детали описаны в разделе 8.3), Алиса (соответственно Боб) может быть уверена, что общий сеансовый ключ, вычисленный на основе числа g^{ab}, является случайным, поскольку она использовала случайный показатель степени. Это возможно благодаря тому, что отображения $g^b \mapsto \left(g^b\right)^a$ и $g^b \mapsto (g^a)^b$ являются перестановками в группе. Следовательно, если случайный показатель степени не превышает порядок группы, то число g^a (соответственно число g^b) отображается в случайный элемент группы g^{ab}.

Теперь покажем, как снять второе ограничение. Заметим, что гибридная схема, использующая протокол обмена ключами Диффи–Хеллмана, обладает свойством заблаговременной секретности, если Алиса и Боб принимают меры предосторожности, приведенные в разделе 8.3. Для этого Алиса и Боб должны обменяться

сеансовым ключом g^{ab}, а затем стереть показатели a и b до окончания протокола. Кроме того, Алиса и Боб должны уничтожить сеансовый ключ после окончания сеанса связи и правильно разместить исходные сообщения, которыми они обменивались. Если они выполняют эти вполне *стандартные* процедуры, то никакими мерами принуждения перехватчик не сможет взломать исходные сообщения, которыми обменивались Алиса и Боб. Криптоаналитики также не справятся с этой задачей, поскольку свойство заблаговременной секретности (благодаря протоколу обмена ключами Диффи–Хеллмана) является следствием неразрешимости вычислительной проблемы Диффи–Хеллмана (см. раздел 8.4).

В заключение отметим, что можно разработать гибридную схему шифрования, обладающую **доказуемой стойкостью** (provable security) в смысле сильной стойкости. Такие схемы рассматриваются в главе 15.

8.16 Организация канала для обмена открытыми ключами

Атака на протокол обмена ключами Диффи–Хеллмана под названием “человек посередине” (man-in-the-middle attack) — обычный способ взлома криптосистем с открытыми ключами (раздел 8.3.1). Как правило, чтобы переслать секретную информацию, зашифрованную с помощью открытого ключа, отправитель сначала должен убедиться, что сообщение будет доставлено по назначению. Аналогично при получении симметричного ключа шифрования, содержащегося в цифровом конверте, получатель сначала должен убедиться, что он отправлен кем положено.

Итак, независимо от популярности криптографических систем с открытым ключом, необходимо организовать секретный канал связи. Однако в криптографии с открытым ключом выполняется условие $k\ell \neq kd$ (см. рис. 7.1). Следовательно, передача ключа шифрования отправителю сообщения не обязательно связана с раскрытием секретной информации. Таким образом, организация секретного канала для обмена ключами сводится исключительно к задаче аутентификации.

Организация каналов для обмена аутентичными открытыми ключами обсуждается в главе 13. В разделе 13.2 описывается метод организации канала для обмена открытыми ключами на основе каталогов, а в разделе 13.3 — методы личностной криптографии с открытым ключом.

8.17 Резюме

В главе рассмотрено несколько хорошо известных схем шифрования с помощью открытого ключа: протокол обмена ключами Диффи–Хеллмана, а также алгоритмы шифрования RSA, Рабина и Эль-Гамаля. Описаны задачи, неразрешимость которых обусловливает стойкость этих криптосистем.

Стойкость, которая рассматривается в этой главе, основана на принципе "все или ничего", а атаки считаются пассивными. По этой причине все алгоритмы шифрования являются учебными и могут применяться в реальных приложениях только при условии, что все передаваемые данные являются абсолютно случайными, а злоумышленники не ведут себя агрессивно (т.е. не организовывают активных атак). Нестойкость учебных криптосистем, описанных в главе, продемонстрирована на примере разнообразных атак.

Выявлена необходимость более строгого понятия стойкости, которое было бы применимо к реальным криптосистемам. Однако эти системы будут описаны в части V. Читатели, не собирающиеся внимательно изучать часть V, должны тщательно разобрать все атаки, описанные в главе, особенно если они планируют использовать учебные криптосистемы.

Упражнения

8.1. Какими двумя замечательными свойствами обладают учебные криптографические алгоритмы?

8.2. Режим сцепления блоков (CBC) блочного шифра, описанный в разделе 7.8.2, имеет случайные входные данные, и, следовательно, любая утечка информации об исходном тексте исключена. Можно ли считать режим CBC учебным криптографическим алгоритмом? Почему?

8.3. Допустим, что Злоумышленник, организовавший атаку "человек посередине" на протокол обмена ключами Диффи–Хеллмана, только передает сообщения, которыми обмениваются Алиса и Боб (т.е. "человек посередине" не изменяет сообщения, а просто выполняет расшифровку и шифрование с помощью ключей, распределенных между Алисой и Бобом). Какой является эта атака: активной или пассивной?

8.4. В качестве общепринятой нижней границы для размера конечного поля F_q используется число $|q| = 1024$, а в суэкспоненциальном выражении $\mathrm{sub\_exp}(q)$ (8.4.2) число c должно быть меньше двух. Докажите, что при этих условиях существует полиномиальный алгоритм дискретного логарифмирования в поле F_q, причем время работы полиномиального алгоритма решения ограничено полиномом девятой степени, зависящим от размера поля q.

8.5. Допустим, что группа $\langle g \rangle$ имеет несекретный порядок $\mathrm{ord}(g)$. Является ли трудноразрешимой следующая задача? По заданному числу g^c найти числа g^a и g^b, такие что $ab \equiv c (\mathrm{mod}\, \mathrm{ord}(g))$, т.е. создать кортеж Диффи–Хеллмана (g, g^a, g^b, g^c) по заданной паре (g, g^c).

8.6. Как связаны между собой задача дискретного логарифмирования и вычислительная проблема Диффи–Хеллмана?

8.7. Почему в наборе параметров открытого ключа (e, N) криптосистемы RSA показатель степени e должен быть взаимно простым с числом $\phi(N)$?

8.8. Разложение нечетного составного целого числа на множители считается трудноразрешимой задачей. Является ли трудноразрешимой задача о факторизации степени простого числа? (Степенью простого числа называется число $N = p^i$, где p — простое число, а i — целое число. Выполните разложение числа N на простые множители.)

Подсказка: сколько индексов i, где $i > 1$, необходимо перебрать для того, чтобы найти корень i-й степени числа N?

8.9. Если число N является степенью простого числа, одним из методов вычисления корня i-й степени числа N является метод бинарного поиска. Разработайте бинарный алгоритм поиска для вычисления корня степени p^i, где показатель i известен. Докажите эффективность этого алгоритма.

Подсказка: рассмотрите алгоритм бинарного поиска простых чисел, состоящих из $\frac{\log_2}{i}$ битов.

8.10. В криптосистеме RSA функция шифрования является перестановкой в мультипликативной группе по модулю RSA. По этой причине функция RSA называется однонаправленной функцией перестановки с секретом. Является ли функция шифрования в криптосистеме Рабина (Эль-Гамаля) однонаправленной функцией перестановки с секретом?

8.11. Допустим, что $N \approx 2^{1024}$, а элементы из группы $\mathbb{Z}_N^*$ извлекаются случайным образом. Какова вероятность того, что выборочное значение будет меньше, чем 2^{64}? Используя этот результат, докажите, что в алгоритмах шифрования RSA, Рабина и Эль-Гамаля 64-битовый случайный пароль не должен считаться случайным исходным текстом.

8.12. При каких условиях функцию шифрования в криптосистеме Эль-Гамаля можно считать детерминированным алгоритмом?

8.13. Опишите атаки CPA, CCA и CCA2.

8.14. При описании стойкости криптосистем RSA и Рабина к атаке CPA использовался принцип “все или ничего” (теоремы 8.1 и 8.2.1 соответственно). Зачем это нужно?

8.15. Почему любой алгоритм шифрования в криптосистемах с открытым ключом (даже учебный) должен быть устойчивым к атаке CPA?

8.16. Назовите основную причину уязвимости учебных криптографических алгоритмов для активных атак.

8.17. Что такое услуги оракула? Необходимы ли атакующему услуги оракула для расшифровки сообщений в криптосистеме с открытым ключом?

8.18. Поскольку все учебные алгоритмы уязвимы для активных атак, необходимо быть бдительным и не предоставлять услуги оракула для расшифровки сообщений. Насколько практична такая стратегия?

8.19. Активная атака, как правило, сопровождается изменением зашифрованного сообщения, передаваемого по сети. Эффективна ли активная атака, если алгоритм шифрования в криптосистеме с открытым ключом содержит механизм, защищающий целостность данных и распознающий несанкционированные изменения зашифрованных текстов?

8.20. Какие преимущества имеют гибридные системы?