СЕРГЕЙ ПАНАСЕНКО

bhv

АЛГОРИТМЫ ШИФРОВАНИЯ

специальный справочник

Классификация алгоритмов шифрования и методов их вскрытия

Новейшая история симметричного шифрования

Описание алгоритмов

Сергей Панасенко

АЛГОРИТМЫ ШИФРОВАНИЯ

специальный справочник

Санкт-Петербург

«БХВ-Петербург»

2009

УДК 681.3.06
ББК 32.973.26-018.2
П16

Панасенко С. П.

П16 Алгоритмы шифрования. Специальный справочник. — СПб.: БХВ-Петербург, 2009. — 576 с.: ил.

ISBN 978-5-9775-0319-8

Книга посвящена алгоритмам блочного симметричного шифрования. Дана общая классификация криптографических алгоритмов. Рассмотрено более 50 алгоритмов шифрования: история создания и использования, основные характеристики и структура, достоинства и недостатки. Описаны различные виды криптоаналитических атак на алгоритмы шифрования и на их реализации в виде программных или аппаратных шифраторов. Рассказано о конкурсах по выбору стандартов шифрования США и Евросоюза.

Для специалистов в области информационных технологий, преподавателей, студентов и аспирантов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Ирина Иноземцева*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.10.08.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 46,44.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0319-8

Оглавление

Введение

Те или иные способы защиты информации использовались людьми на протяжении тысячелетий. Но именно в течение нескольких последних десятилетий криптография — наука о защите информации — переживает невиданный доселе прогресс, обусловленный, как минимум, двумя важными факторами:

- бурное развитие вычислительной техники и ее повсеместное использование привело к тому, что теперь в подавляющем большинстве случаев криптография защищает именно компьютерную информацию;
- тогда как раньше криптография была уделом государственных структур, сейчас криптографические методы защиты могут использовать (и используют) обычные люди и организации, хотя бы для защиты своей собственной переписки от посторонних глаз.

То же самое касается и разработки криптографических алгоритмов — известно множество алгоритмов шифрования, и далеко не все из них разработаны «в недрах спецслужб» или научными институтами — встречаются весьма удачные и широко используемые алгоритмы, разработанные частными лицами.

Эта книга посвящена алгоритмам шифрования, а именно, алгоритмам блочного симметричного шифрования, которые, на взгляд автора, являются наиболее широко используемыми в современном компьютерном мире. В книге собраны описания более 50 алгоритмов шифрования (и еще около ста их вариантов), которые автор считает наиболее широко используемыми, наиболее удачными, обладающими наиболее полезными свойствами или просто наиболее интересными по своей структуре или истории.

Как известно, шифрование решает одну из основных проблем защиты информации — а именно, проблему обеспечения конфиденциальности данных. Известно также, что во все времена предпринимались усилия получить именно ту информацию, которая по той или иной причине защищена от посторонних. Поэтому немалая часть книги посвящена криптоаналитическим атакам,

т. е. методам, с помощью которых можно попытаться вскрыть зашифрованную информацию.

Книга состоит из трех глав и приложения.

В *главе 1* приведена общая классификация криптографических алгоритмов, описана структура алгоритмов шифрования и наиболее часто используемые режимы их применения. Там же описаны различные виды криптоаналитических атак на алгоритмы шифрования и на их реализации в виде программных или аппаратных шифраторов.

Глава 2 посвящена краткому обзору новейшей истории алгоритмов шифрования. Она описывает прошедшие в течение последнего десятилетия конкурсы по выбору стандартов шифрования США и Евросоюза. Именно эти конкурсы, по мнению автора, очень сильно повлияли на современное развитие криптографии и криптоанализа.

Глава 3 содержит описание алгоритмов шифрования, т. е., в основном, следующие сведения:

- ❒ историю создания и использования алгоритмов;
- ❒ основные характеристики и структуру алгоритмов, включая подробное описание используемых в алгоритмах преобразований;
- ❒ достоинства и недостатки алгоритмов;
- ❒ описание известных методов вскрытия алгоритмов.

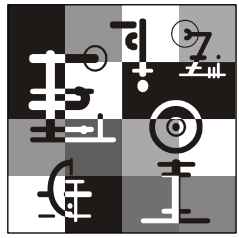
Алгоритмы снабжены схемами, существенно упрощающими понимание структуры конкретного алгоритма. Стоит отметить, что описание большинства алгоритмов достаточно подробно для разработки на их основе программного или аппаратного шифратора.

В *Приложение 1* вынесены громоздкие таблицы, нахождение которых в основном тексте помешало бы его восприятию читателями.

Стоит сказать и о том, что при написании книги не использована какая-либо информация ограниченного пользования — абсолютно все источники информации об алгоритмах (за исключением ряда общедоступных книг и журнальных статей) найдены автором на открытых ресурсах сети Интернет.

Автор выражает благодарность своим коллегам по работе в ООО «АНКАД», многие из которых оказывали автору различное содействие при создании этой книги.

Автор будет признателен читателям за любые замечания по этой книге, в том числе критические. Жду ваших писем по адресу **serg@panasenko.ru**.



Глава 1

Классификация алгоритмов шифрования и методов их вскрытия

В этой главе дана общая классификация криптографических алгоритмов, приведены структуры алгоритмов шифрования и наиболее часто используемые режимы их применения. Здесь же описаны различные виды криптоаналитических атак на алгоритмы шифрования и на их реализации в виде программных или аппаратных шифраторов.

1.1. Криптографические алгоритмы

Для начала обсудим, какие же типы криптографических алгоритмов используются для защиты информации. Криптоалгоритмы, прежде всего, делятся на три категории (рис. 1.1) [7, 263]:

- *бесключевые* алгоритмы, которые не используют каких-либо ключей в процессе криптографических преобразований;
- *одноключевые* алгоритмы, использующие в своих вычислениях некий секретный ключ;
- *двухключевые* алгоритмы, в которых на различных этапах вычислений применяются два вида ключей: секретные и открытые.

Рассмотрим вкратце основные типы криптоалгоритмов.

- *Хэш-функции*. Выполняют «свертку» данных переменной длины в последовательность фиксированного размера — фактически это контрольное суммирование данных, которое может выполняться как с участием некоего ключа, так и без него. Такие функции имеют весьма широкое применение в области защиты компьютерной информации, например:
  - для подтверждения целостности любых данных в тех случаях, когда использование электронной подписи (см. далее) невозможно (например, из-за большой ресурсоемкости) или является избыточным;

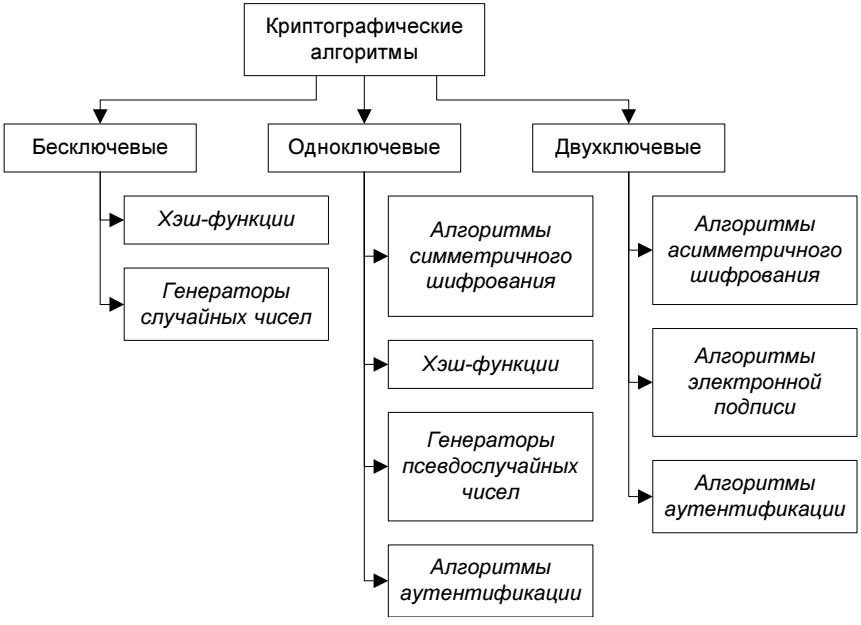


Рис. 1.1. Классификация криптографических алгоритмов

- в самих схемах электронной подписи — подписывается обычно хэш данных, а не все данные целиком;
- в различных схемах аутентификации пользователей (см. далее).

❒ *Генераторы случайных чисел*. Случайные числа необходимы, в основном, для генерации секретных ключей шифрования, которые, в идеале, должны быть абсолютно случайными. Нужны они и для вычисления электронной цифровой подписи, и для работы многих алгоритмов аутентификации.

❒ *Алгоритмы симметричного шифрования* — алгоритмы шифрования, в которых для зашифровывания и расшифровывания используется один и тот же ключ, или ключ расшифровывания легко вычисляется из ключа зашифровывания и наоборот.

Симметричное шифрование бывает двух видов: блочное и потоковое.

- *Блочное шифрование* — в этом случае информация разбивается на блоки фиксированной длины (например, 64 или 128 битов), после чего эти блоки поочередно шифруются. Причем в различных алгоритмах шифрования или даже в разных режимах работы одного и того же алгоритма блоки могут шифроваться независимо друг от друга или «со сцеплением» — когда результат зашифровывания текущего блока данных

зависит от значения предыдущего блока или от результата зашифровывания предыдущего блока (*см. разд. 1.4*). Данная книга посвящена описанию криптоалгоритмов именно этой, наиболее обширной, категории алгоритмов шифрования.

- *Потоковое шифрование* — необходимо, прежде всего, в тех случаях, когда информацию невозможно разбить на блоки — скажем, некий поток данных, каждый символ которых должен быть зашифрован и отправлен куда-либо, не дожидаясь остальных данных, достаточных для формирования блока. Поэтому алгоритмы потокового шифрования шифруют данные побитно или посимвольно. Хотя стоит сказать, что некоторые классификации не разделяют блочное и потоковое шифрование, считая, что потоковое шифрование — это шифрование блоков единичной длины [263].

- *Генераторы псевдослучайных чисел*. Не всегда возможно получение абсолютно случайных чисел — для этого необходимо наличие качественных аппаратных генераторов. Однако на основе алгоритмов симметричного шифрования можно построить очень качественный генератор псевдослучайных чисел.

- *Алгоритмы аутентификации*. Позволяют проверить, что пользователь (или удаленный компьютер) действительно является тем, за кого себя выдает. Простейшая схема аутентификации — парольная — не требует наличия каких-либо криптографических ключей, но доказанно является слабой. А с помощью секретного ключа можно построить заметно более сильные схемы аутентификации. Пример аутентификации пользователя сервером (рис. 1.2):

  Этап 1. Сервер генерирует случайное число

  Этап 2. и отправляет его пользователю.

  Этап 3. Пользователь зашифровывает полученное число секретным ключом и отправляет результат серверу.

  Этап 4. Сервер расшифровывает полученные данные таким же секретным ключом

  Этап 5. и сравнивает с исходным числом.

  Равенство чисел означает, что пользователь обладает требуемым секретным ключом, т. е. ему удалось доказать свою легитимность.

- *Алгоритмы асимметричного шифрования*. Применяют два вида ключей: открытый ключ для зашифровывания информации и секретный — для расшифровывания. Секретный и открытый ключи связаны между собой достаточно

сложным соотношением, главное в котором — легкость вычисления открытого ключа из секретного и невозможность (за ограниченное время при реальных ресурсах) вычисления секретного ключа из открытого при достаточно большой размерности операндов. Любая информация, зашифрованная общедоступным открытым ключом, может быть расшифрована только обладателем секретного ключа, из которого был вычислен данный открытый (рис. 1.3):

Этап 1. Пользователь B зашифровывает сообщение на открытом ключе пользователя A (который когда-либо передал его пользователю B).

Этап 2. Пользователь A расшифровывает сообщение своим секретным ключом.

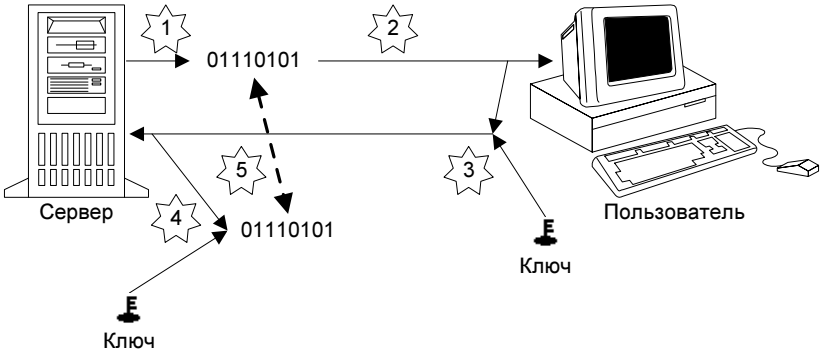


Рис. 1.2. Пример аутентификации пользователя сервером

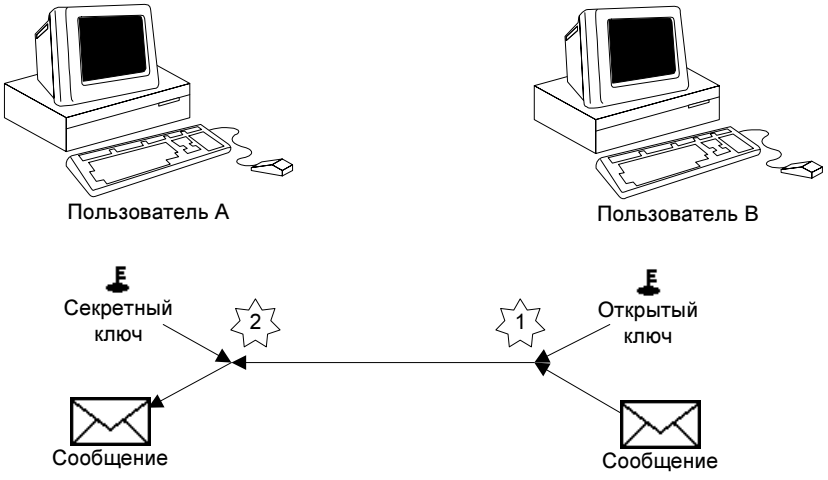


Рис. 1.3. Асимметричное шифрование

- *Алгоритмы электронной подписи*. Используют секретный ключ для вычисления электронной цифровой подписи данных, а вычисляемый из него открытый — для ее проверки.

  Как и асимметричное шифрование, это двухключевые алгоритмы с таким же простым вычислением открытого ключа из секретного и практической невозможностью обратного вычисления. Однако назначение ключей является совершенно другим:

  - секретный ключ используется для вычисления электронной подписи;
  - открытый ключ необходим для ее проверки.

  При соблюдении безопасного хранения секретного ключа, никто, кроме его владельца, не в состоянии вычислить верную электронную подпись какого-либо электронного документа.

1.2. Категории алгоритмов шифрования

Шифрование — это основной криптографический метод защиты информации, обеспечивающий ее конфиденциальность [14, 21].

Шифрование информации — это преобразование открытых данных в зашифрованные и наоборот. Первая часть этого процесса называется зашифровыванием, вторая — расшифрованием.

Шифрование можно представить в виде следующих формул:

$$C = E_{k1}(M);$$
$$M' = D_{k2}(C).$$

где:

- M (message) — открытая информация, в литературе по защите информации часто обозначается словосочетанием *открытый текст*;
- C (cipher text) — полученный в результате зашифровывания *шифртекст* (или *криптограмма*);
- E (encryption) — функция зашифровывания, выполняющая криптографические преобразования над исходным текстом;
- $k1$ (key) — параметр функции E, называемый *ключом зашифровывания*;
- M' — информация, полученная в результате расшифровывания;
- $k2$ — ключ, с помощью которого выполняется расшифровывание информации;
- D (decryption) — функция расшифровывания, выполняющая обратные зашифровыванию криптографические преобразования над шифртекстом.

В стандарте ГОСТ 28147-89 (*см. разд. 3.1*) понятие «ключ» определено следующим образом: «Конкретное секретное состояние некоторых параметров алгоритма криптографического преобразования, обеспечивающее выбор одного преобразования из совокупности всевозможных для данного алгоритма преобразований» [4]. Иными словами, ключ является уникальным элементом, с помощью которого можно варьировать результат работы алгоритма шифрования: один и тот же открытый текст при использовании различных ключей будет зашифрован по-разному.

Для того чтобы результат операций зашифровывания и последующего расшифровывания совпал с исходным сообщением (т. е. для получения $M' = M$), необходимо одновременное выполнение двух условий:

- функция расшифровывания D должна соответствовать функции зашифровывания E;
- ключ расшифровывания $k2$ должен соответствовать ключу зашифровывания $k1$.

При отсутствии верного ключа $k2$ получить исходное сообщение $M' = M$ невозможно, если для зашифровывания использовался криптографически стойкий алгоритм шифрования. Понятие *криптостойкости* описано в *разд. 1.5*.

Алгоритмы шифрования можно разделить на две категории.

- Алгоритмы симметричного шифрования. Определяются следующим соотношением ключей зашифровывания и расшифровывания:

$$k1 = k2 = k,$$

  т. е. функциями E и D используется один и тот же ключ шифрования.

- Алгоритмы асимметричного шифрования, в которых ключ зашифровывания $k1$ вычисляется из ключа $k2$ таким образом, что обратное вычисление невозможно; например, по следующей формуле:

$$k1 = a^{k2} \bmod p,$$

  где a и p — параметры используемого алгоритма шифрования.

1.3. Структура алгоритмов симметричного шифрования

Подавляющее большинство современных алгоритмов шифрования работают весьма схожим образом: над шифруемым текстом выполняется некое преобразование с участием ключа шифрования, которое повторяется определенное число раз (раундов). При этом по виду повторяющегося преобразования алгоритмы шифрования принято делить на несколько категорий. Здесь также

существуют различные классификации, приведу одну из них. Итак, по своей структуре алгоритмы шифрования классифицируются следующим образом [9, 50, 187, 284, 395].

Алгоритмы на основе сети Фейстеля

Сеть Фейстеля (Feistel network) подразумевает разбиение обрабатываемого блока данных на несколько субблоков (чаще всего — на два), один из которых обрабатывается некоей функцией *f* и накладывается на один или несколько остальных субблоков. На рис. 1.4 приведена наиболее часто встречающаяся структура алгоритмов на основе сети Фейстеля.

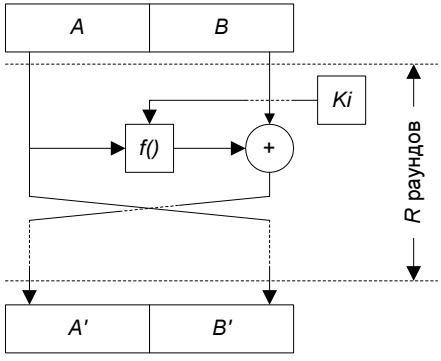


Рис. 1.4. Сеть Фейстеля

Дополнительный аргумент функции *f*, обозначенный на рис. 1.4 как K_i, называется *ключом раунда*. Ключ раунда является результатом обработки ключа шифрования процедурой расширения ключа, задача которой — получение необходимого количества ключей K_i из исходного ключа шифрования относительно небольшого размера (в настоящее время достаточным для ключа симметричного шифрования считается размер 128 битов). В простейших случаях процедура расширения ключа просто разбивает ключ на несколько фрагментов, которые поочередно используются в раундах шифрования; существенно чаще процедура расширения ключа является достаточно сложной, а ключи K_i зависят от значений большинства битов исходного ключа шифрования.

Наложение обработанного субблока на необработанный чаще всего выполняется с помощью логической операции «исключающее или» (Exclusive OR, XOR), как показано на рис. 1.4. Достаточно часто вместо XOR здесь используется сложение по модулю 2^n, где n — размер субблока в битах. После наложения субблоки меняются местами, т. е. в следующем раунде алгоритма обрабатывается уже другой субблок данных.

Такая структура алгоритмов шифрования получила свое название по имени Хорста Фейстеля (Horst Feistel) — одного из разработчиков алгоритма шифрования Lucifer (*см. разд. 3.31*) и разработанного на его основе алгоритма DES (Data Encryption Standard) — бывшего (но до сих пор широко используемого) стандарта шифрования США (*см. разд. 3.15*). Оба этих алгоритма имеют структуру, аналогичную показанной на рис. 1.4. Среди других алгоритмов, основанных на сети Фейстеля, можно привести в пример отечественный стандарт шифрования ГОСТ 28147-89 (*см. разд. 3.1*), а также другие весьма известные алгоритмы: RC5 (*см. разд. 3.42*), Blowfish (*см. разд. 3.8*), TEA (*см. разд. 3.55*), CAST-128 (*см. разд. 3.10*) и т. д.

На сети Фейстеля основано большинство современных алгоритмов шифрования — благодаря множеству преимуществ подобной структуры, среди которых стоит отметить следующие:

- алгоритмы на основе сети Фейстеля могут быть сконструированы таким образом, что для зашифровывания и расшифровывания может использоваться один и тот же код алгоритма — разница между этими операциями может состоять лишь в порядке применения ключей K_i; такое свойство алгоритма наиболее полезно при его аппаратной реализации или на платформах с ограниченными ресурсами; в качестве примера такого алгоритма можно привести ГОСТ 28147-89;
- алгоритмы на основе сети Фейстеля являются наиболее изученными — таким алгоритмам посвящено огромное количество криптоаналитических исследований, что является несомненным преимуществом как при разработке алгоритма, так и при его анализе.

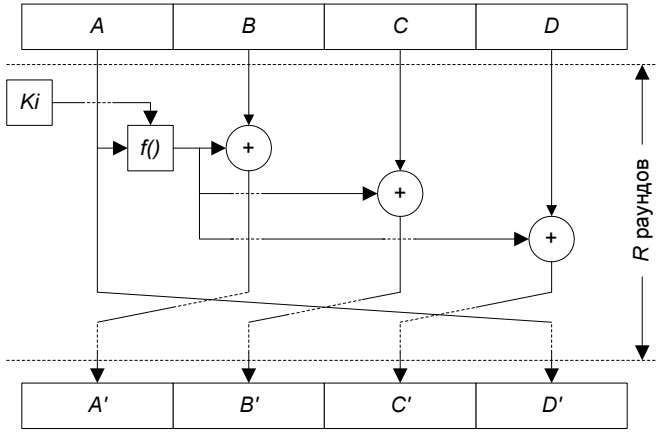


Рис. 1.5. Расширенная сеть Фейстеля

Существует и более сложная структура сети Фейстеля, пример которой приведен на рис. 1.5. Такая структура называется *обобщенной* или *расширенной* сетью Фейстеля и используется существенно реже традиционной сети Фейстеля. Примером такой сети Фейстеля может служить алгоритм RC6 (*см. разд. 3.43*).

Алгоритмы на основе подстановочно-перестановочных сетей

В отличие от сети Фейстеля, *SP-сети* (Substitution-permutation network, подстановочно-перестановочная сеть) обрабатывают за один раунд целиком шифруемый блок. Обработка данных сводится, в основном, к заменам (когда, например, фрагмент входного значения заменяется другим фрагментом в соответствии с таблицей замен, которая может зависеть от значения ключа K_i) и перестановкам, зависящим от ключа K_i (упрощенная схема показана на рис. 1.6). Впрочем, такие операции характерны и для других видов алгоритмов шифрования, поэтому, на мой взгляд, название «подстановочно-перестановочная сеть» является достаточно условным.

SP-сети являются гораздо менее распространенными, чем сети Фейстеля; в качестве примера SP-сетей можно привести алгоритмы Serpent (*см. разд. 3.48*) или SAFER+ (*см. разд. 3.45*).

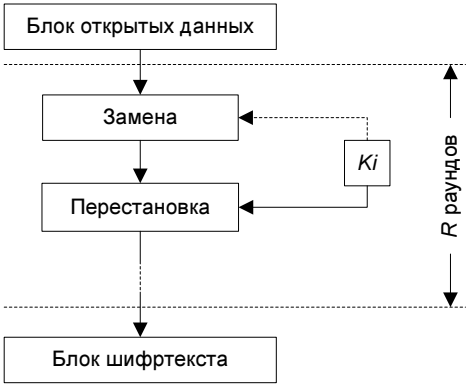


Рис. 1.6. SP-сеть

Алгоритмы со структурой «квадрат»

Для структуры «*квадрат*» (Square) характерно представление шифруемого блока данных в виде двумерного байтового массива. Криптографические преобразования могут выполняться над отдельными байтами массива, а также над его строками или столбцами.

Эта структура получила свое название от алгоритма Square (*см. разд. 3.54*), который был разработан в 1996 г. Винсентом Риджменом (Vincent Rijmen) и Джоан Деймен (Joan Daemen) — будущими авторами алгоритма Rijndael (*см. разд. 3.3*), ставшего в США новым стандартом шифрования AES после победы на открытом конкурсе (*см. разд. 2.1*). Алгоритм Rijndael также имеет Square-подобную структуру; кроме того, в качестве примера можно привести алгоритмы SHARK (более ранняя разработка Риджмена и Деймен, *см. разд. 3.50*) и Crypton (*см. разд. 3.12*). Недостатком алгоритмов со структурой «квадрат» является их недостаточная изученность, что не помешало алгоритму Rijndael стать новым стандартом шифрования США.

Алгоритмы с нестандартной структурой

Изобретательность безгранична, поэтому как-либо классифицировать все возможные варианты алгоритмов шифрования представляется сложным, т. е. существуют такие алгоритмы, которые невозможно причислить ни к одному из перечисленных типов. В качестве примера алгоритма с нестандартной структурой можно привести уникальный по своей структуре алгоритм FROG (*см. разд. 3.19*), в каждом раунде которого по достаточно сложным правилам выполняется модификация двух байтов шифруемых данных.

Строгие границы между описанными выше структурами не определены, поэтому достаточно часто встречаются алгоритмы, причисляемые различными экспертами к разным типам структур. Например, алгоритм CAST-256 (*см. разд. 3.11*) относится его автором к SP-сети, а многими экспертами называется расширенной сетью Фейстеля. Другой пример — алгоритм HPC (*см. разд. 3.23*), называемый его автором сетью Фейстеля, но относимый экспертами к алгоритмам с нестандартной структурой [50, 187].

1.4. Режимы работы алгоритмов

В 1980 г. в США был принят стандарт [151], определяющий режимы работы алгоритма DES — стандарта шифрования США (*см. разд. 3.15*). Можно сказать, что этот стандарт уточнял подробности реализации DES для различных применений.

В стандарте были определены следующие режимы работы алгоритма DES:

- электронная кодовая книга ECB (Electronic Code Book);
- сцепление блоков шифра CBC (Cipher Block Chaining);
- обратная связь по шифртексту CFB (Cipher Feed Back);
- обратная связь по выходу OFB (Output Feed Back).

Рассмотренные далее режимы не привязаны к конкретному алгоритму — фактически любые алгоритмы блочного симметричного шифрования могут быть использованы (и используются) в данных режимах работы.

Электронная кодовая книга

Наиболее простым из них является режим ECB. Суть его состоит в том (рис. 1.7), что каждый блок шифруемых данных «прогоняется» через алгоритм шифрования отдельно и независимо от других блоков:

$$C_i = E_k(M_i),$$

где:

- E_k — функция зашифровывания на ключе k;
- M_i и C_i — блоки открытого текста и соответствующие им блоки шифртекста.

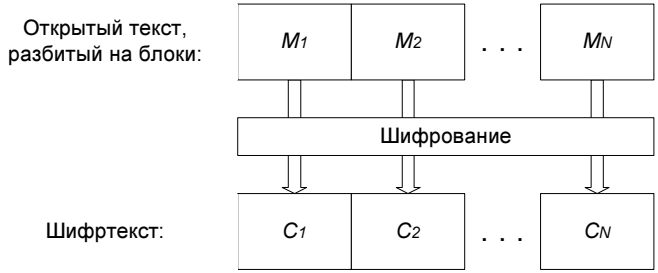


Рис. 1.7. Режим ECB

Аналогично выполняется и расшифровывание — блоки шифртекста обрабатываются поочередно и независимо.

Режим ECB плох тем, что если открытый текст содержит какое-либо количество блоков с одинаковым содержимым (например, большой массив, проинициализированный нулевыми или единичными битами), то и шифртекст будет содержать такое же количество одинаковых блоков. Это непозволительно, поскольку дает криптоаналитику информацию о структуре зашифрованной информации, что может существенно облегчить вскрытие алгоритма шифрования (т. е. получение открытого текста из зашифрованного или вычисление ключа шифрования). Поэтому режим ECB должен использоваться только для шифрования ключей друг на друге в многоключевых схемах [14]. Допускается также шифрование небольших фрагментов данных при условии их неповторяемости [263].

Есть и еще один недостаток — в режиме ECB алгоритм шифрует данные только поблочно. При необходимости, например, зашифровать алгоритмом DES 8 битов придется дополнить эти данные до 64-битного размера блока, зашифровать блок целиком, а при расшифровывании отбросить дополняющие биты. Такое не всегда возможно.

Достоинство режима ECB — простота реализации по сравнению с другими режимами. Однако, по словам известного криптолога Брюса Шнайера (Bruce Schneier), «из-за своей простоты в большинстве существующих коммерческих программ используется режим ECB, хотя этот режим наиболее уязвим к вскрытию» [28].

Сцепление блоков шифра

Существенно более стойким является режим CBC. В этом режиме (рис. 1.8) перед шифрованием каждого *i*-го блока шифруемых данных выполняется его побитовое сложение по модулю 2 с результатом шифрования предыдущего, $(i-1)$-го, блока. То есть для режима CBC шифрование выглядит так:

$$C_i = E_k(M_i \oplus C_{i-1}).$$

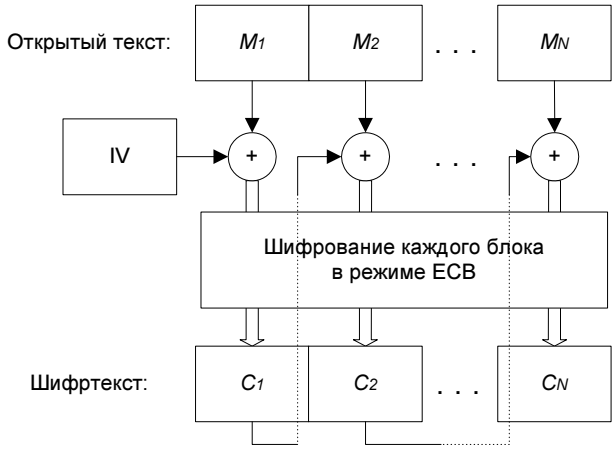


Рис. 1.8. Режим CBC

Видно, что результат шифрования каждого блока зависит не только от содержимого шифруемого блока, но и от всех предыдущих блоков открытого текста. При шифровании первого блока открытого текста вместо результата

зашифровывания предыдущего блока используется *вектор инициализации* (Initialization Vector, IV):

$$C_1 = E_k(M_1 \oplus IV).$$

Варьируя значение вектора инициализации, можно получать различные шифртексты для одинаковых открытых текстов. Естественно, при расшифровывании шифртекста, полученного в режиме CBC, вектор инициализации должен быть известен. Это справедливо и для рассмотренных далее режимов CFB и OFB.

Аналогично предыдущему режиму, данные, размер которых меньше 64-битного блока, придется перед обработкой дополнять до 64 битов.

Режим CBC используется непосредственно для зашифровывания данных, в том числе в больших объемах. Кроме того, последний блок шифртекста C_N может использоваться для контроля целостности сообщений, поскольку его значение зависит от содержимого всех блоков открытого текста, вектора инициализации и ключа:

$$C_N = f(M_1, M_2, ...M_N, IV, k).$$

Обратная связь по шифртексту

Режим CFB более сложен в реализации, чем два предыдущих. Шифрование данных в этом режиме выполняется следующим образом (рис. 1.9):

1. Вектор инициализации IV записывается в регистр 1.
2. 64-битное содержимое регистра 1 зашифровывается, результат помещается в регистр 2.
3. Из регистра 2 выбираются L левых битов ($1 \le L \le 64$), которые накладываются операцией XOR на L-битный блок шифруемого текста. Результат этого шага — L-битный блок шифртекста.
4. Содержимое регистра 1 сдвигается влево на L битов.
5. Регистр 1 дополняется справа L-битным блоком шифртекста. При необходимости продолжить шифрование данных шаги 2–5 повторяются в цикле до обработки всех шифруемых данных.

Фактически в режиме CFB алгоритм шифрования на основе вектора инициализации, ключа шифрования и предыдущих блоков шифртекста генерирует псевдослучайную последовательность, которая накладывается на открытый текст. Поскольку операция XOR обладает следующим свойством:

$$(x \oplus y) \oplus y = x$$

для любых значений x и y, то для расшифровывания следует сгенерировать такую же последовательность. Не имея ключа и значения вектора инициализации, такую же последовательность сгенерировать невозможно.

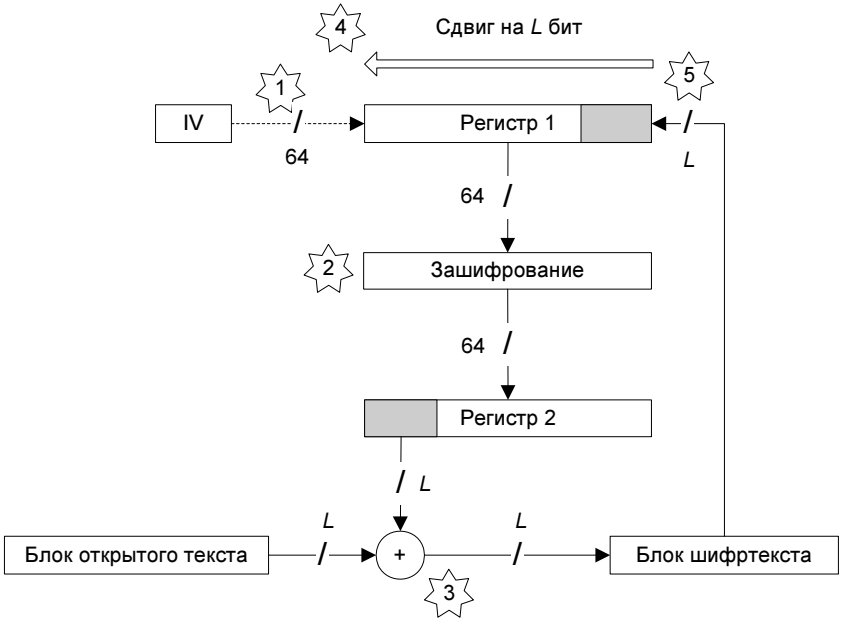


Рис. 1.9. Режим CFB

Обратная связь по выходу

Режим шифрования OFB похож на предыдущий (рис. 1.10):

1. Вектор инициализации IV записывается в регистр 1.
2. Содержимое регистра 1 зашифровывается, результат помещается в регистры 1 и 2.
3. Из регистра 2 выбираются L левых битов, которые накладываются операцией XOR на L-битный блок шифруемого текста, в результате получается L-битный блок шифртекста. При необходимости продолжить шифрование шаги 2 и 3 повторяются в цикле.

Здесь приведена более поздняя версия режима OFB (описанная в стандарте ISO 10116); согласно [151] в регистр 1 «возвращался» не полностью 64-битный результат зашифровывания регистра 1, а младшие L битов (что больше напоминает режим CFB); исходный вариант режима OFB считается менее криптографически стойким, чем модифицированный [28, 263].

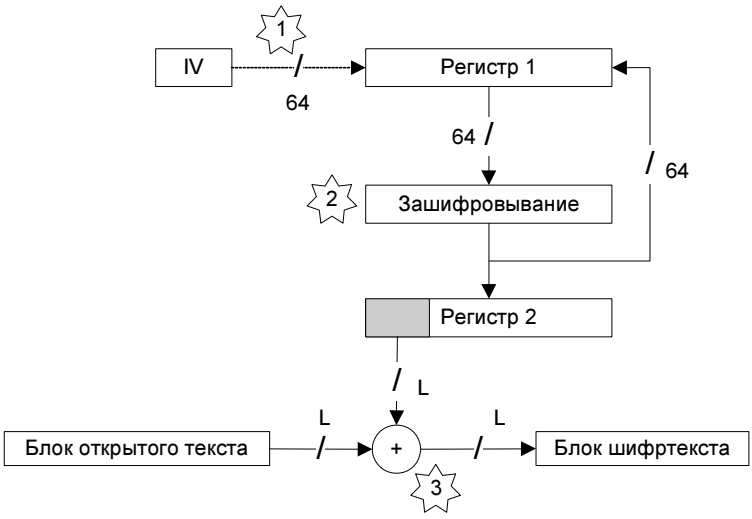


Рис. 1.10. Режим OFB

Существует и более простой режим OFB — *режим счетчика* [263], в котором обратная связь вообще отсутствует, а содержимое регистра 1 перед каждым его зашифровыванием увеличивается на 1.

Аналогично режиму CFB, для расшифровывания необходимо сгенерировать ту же последовательность и наложить ее на шифртекст.

Отличие от предыдущего режима лишь в том, что значение регистра 1 заменяется в цикле его же зашифрованным содержимым. Отсюда проистекает важное свойство режима OFB — накладываемая на шифруемый текст последовательность зависит только от значения вектора инициализации и ключа шифрования. То есть шифрующую последовательность можно сгенерировать заранее и, например, использовать ее в периоды максимальной нагрузки на шифрующий компьютер или устройство, восполняя последовательность в фоновом режиме. Такая возможность весьма важна для серверных компонент распределенных систем, поскольку позволяет существенно увеличить их пиковую производительность.

Еще одно важное отличие режима OFB от остальных состоит в том, что при возникновении ошибки в одном бите шифртекста после расшифровывания возникает ошибка только в одном бите расшифрованных данных, тогда как в остальных режимах ошибочно расшифруются один или два блока.

Стоит сказать и о том, что в режимах OFB и CFB алгоритмы блочного симметричного шифрования можно использовать в качестве потокового шифра (*см. разд. 1.1*), установив L равным размеру символов потока (например, 1 или 8).

Другие режимы работы

Существуют и другие режимы работы, большинство которых являются незначительной модификацией описанных выше и применяются существенно реже [28].

Кроме того, при многократном шифровании используется множество специфических режимов работы. Многократное шифрование подробно рассмотрено на примере алгоритмов Double DES и Triple DES в *разд. 3.15*.

1.5. Атаки на алгоритмы шифрования

Рассмотрим стойкость алгоритмов шифрования к разнообразным криптоаналитическим действиям злоумышленника, направленным на их вскрытие [6, 263]. Эта характеристика является важнейшей для алгоритмов шифрования и называется *криптостойкостью*. Криптостойкость алгоритма обычно измеряется временем, которое необходимо на его вскрытие при неких фиксированных ресурсах, имеющихся в распоряжении у злоумышленника. Например, известно, что если у злоумышленника есть миллион процессоров, каждый из которых может перебрать миллион ключей алгоритма DES в секунду, то на полный перебор всех вариантов 56-битного ключа DES уйдет 20 часов. А это означает, что криптостойкости алгоритма DES недостаточно для защиты каких-либо долговременных данных, но его вполне можно применять для шифрования какой-либо срочной информации, раскрытие которой через те же 20 часов уже не страшно.

Цели атак

Атакуя алгоритм шифрования, злоумышленник может преследовать следующие цели:

- нахождение открытого текста, имея его в зашифрованном виде, но не имея секретного ключа;
- нахождение самого́ секретного ключа.

В первом случае злоумышленнику необходимо какое-либо конкретное зашифрованное сообщение; достигнув же второй цели, т. е. получив секретный ключ, злоумышленник сможет читать все сообщения, зашифрованные на нем, что несравнимо опаснее. Успешное получение злоумышленником секретного ключа называется *полным раскрытием* алгоритма шифрования.

Злоумышленник может иметь целью и нахождение ключа, эквивалентного секретному. *Эквивалентными* называются ключи, которые являются различ-

ными, но приводят к одному и тому же результату шифрования. При успешном нахождении такого ключа злоумышленнику уже не нужен настоящий секретный ключ — все необходимое он расшифрует с помощью эквивалентного.

Стоит сказать и о том, что достижение перечисленных выше целей может оказаться неосуществимым при имеющихся у злоумышленника ресурсах. В этом случае злоумышленник может стремиться к *частичному раскрытию* секретного ключа или открытого сообщения. Частичное раскрытие секретного ключа может, например, помочь злоумышленнику значительно сузить область перебора секретных ключей.

Классификация атак

Атаки на алгоритмы шифрования принято классифицировать в зависимости от того набора информации, который имеет злоумышленник перед осуществлением своей атаки. Прежде всего криптоаналитические атаки можно разделить на две категории.

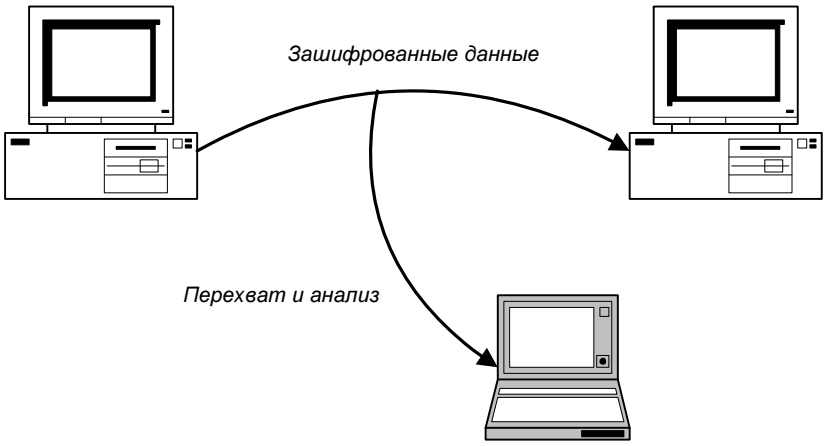


Рис. 1.11. Пассивный перехват зашифрованных данных

Категория 1. У криптоаналитика есть только возможность пассивного прослушивания некоего канала, по которому пересылаются зашифрованные данные (рис. 1.11). В результате у злоумышленника есть лишь набор шифртекстов, зашифрованных на определенном ключе. Такая атака называется атакой *с известным шифртекстом*. Она наиболее сложна, но этот вариант

атаки наиболее распространен, поскольку он является самым «жизненным» — в подавляющем большинстве реальных случаев криптоаналитик не имеет возможности получить больше данных.

Рис. 1.12. Активное воздействие на шифратор

Категория 2. Предполагает, что у криптоаналитика есть некое шифрующее устройство с прошитым ключом шифрования, который и является целью атаки. Таким устройством может быть, например, криптографическая смарт-карта. Криптоаналитик может выполнять с шифратором определенные (допускаемые шифратором и его техническим окружением, а также тактическими условиями осуществления атаки) действия для получения требуемой ему информации, например, «прогонять» через шифратор какие-либо открытые тексты для получения соответствующих им шифртекстов (рис. 1.12). В зависимости от данных, которые криптоаналитик может «добыть» у шифратора, существуют следующие виды атак [28, 263].

- *Атака с известным открытым текстом*. Предполагает наличие у криптоаналитика некоторого количества пар текстов, каждая из которых представляет собой открытый текст и соответствующий ему шифртекст.
- *Атака с выбранным открытым текстом*. У криптоаналитика есть возможность выбора открытых текстов для получения соответствующих им шифртекстов (как это может быть полезно криптоаналитику, будет рассмотрено далее).

- *Адаптивная атака с выбором открытого текста*. Криптоаналитик может не просто выбирать открытые тексты для зашифровывания , но и делать это многократно, с учетом результатов анализа ранее полученных данных.
- *Атака с выбором шифртекста*. Криптоаналитик может выбирать шифртексты и, прогоняя их через шифратор, получать путем расшифровывания соответствующие им открытые тексты.
- *Адаптивная атака с выбором шифртекста*. По аналогии с описанными ранее атаками ясно, что криптоаналитик может многократно выбирать шифртексты для их расшифровывания с учетом предыдущих результатов.

Теоретически возможности криптоаналитика могут и не ограничиваться перечисленными выше; более серьезные варианты воздействия криптоаналитика на шифратор будут рассмотрены в *разд. 1.8*.

Количественная оценка криптостойкости алгоритмов шифрования

Криптостойкость является количественной характеристикой алгоритмов шифрования — для вскрытия конкретного алгоритма шифрования при определенных условиях (в том числе определенным криптоаналитическим методом) требуется определенное количество ресурсов. Ресурсами в данном случае являются:

- количество информации, необходимое для осуществления атаки, — например, количество пар известных или выбранных текстов;
- время, необходимое для осуществления атаки; обычно измеряется в количестве тестовых операций шифрования атакуемым алгоритмом, выполнение которых при соблюдении остальных необходимых условий позволит, например, вычислить ключ шифрования;
- память, необходимая для хранения используемой при атаке информации; также является немаловажной характеристикой, поскольку многие атаки могут предъявлять весьма существенные требования к памяти.

Совокупность этих трех величин характеризует конкретную атаку на конкретный алгоритм шифрования. А лучшая (для которой требуется минимальный набор ресурсов) из возможных атак на алгоритм характеризует его криптостойкость.

Здесь и далее подразумевается, что сам алгоритм шифрования атакующему известен — неизвестен только ключ. Подавляющее большинство криптоаналитических методов (которые будут рассмотрены далее) основаны на доскональном знании криптоаналитиком атакуемого алгоритма. Существует и еще

одна немаловажная характеристика алгоритма шифрования — насколько шифртексты, полученные с его помощью, отличаются от случайной последовательности. Причем данная характеристика может быть выражена количественно в тех же трех описанных выше типах ресурсов.

Криптоанализ модифицированных алгоритмов

Существует немало алгоритмов шифрования, которые являются криптографически стойкими. В известной работе [95] понятие *стойкого* (strong) алгоритма определено так:

- ❒ алгоритм является криптографически стойким, если не существует каких-либо методов его вскрытия, кроме метода «грубой силы» (brute force), который будет рассмотрен далее;
- ❒ кроме того, размер ключа алгоритма является достаточно большим для того, чтобы использование метода «грубой силы» стало невозможным при текущем уровне развития вычислительной техники.

Однако, например, бывает необходимо сравнить между собой два или более криптографически стойких алгоритма шифрования (как, например, на открытом конкурсе по выбору AES, нового стандарта шифрования США — *см. разд. 2.1*). В этом случае используют другую характеристику (скорее качественную, чем количественную) — запас криптостойкости (security margin).

Известно, что подавляющее большинство современных алгоритмов шифрования состоит из определенного количества раундов, в каждом из которых повторяются одни и те же (или схожие) преобразования над шифруемыми данными. Для определения запаса криптостойкости анализируют алгоритм с *усеченным* количеством раундов — т. е. модификацию исследуемого алгоритма, в которой количество раундов уменьшено по сравнению с конкретным предусмотренным в алгоритме количеством раундов. Запас криптостойкости можно определить как соотношение исходного количества раундов исследуемого алгоритма к максимальному количеству раундов его модификаций, не являющихся криптографически стойкими.

Другой вариант определения запаса криптостойкости — анализ модификаций исследуемого алгоритма с незначительными изменениями структуры раунда. Один из наиболее ярких примеров — вскрытие алгоритма Skipjack-3XOR при наличии всего 2^9 выбранных открытых текстов и соответствующих им шифртекстов выполнением всего около миллиона тестовых операций шифрования (*см. разд. 3.52*). По современным меркам, благодаря данной атаке Skipjack-3XOR можно считать весьма слабым алгоритмом, а ведь он отличается от известного и достаточно распространенного алгоритма шифрования

Skipjack всего лишь тем, что из структуры последнего удалены 3 конкретных операции XOR из предусмотренных алгоритмом Skipjack 320 (!) подобных операций. Соответственно, были (однако недоказанные) предположения о недостаточном запасе криптостойкости у алгоритма Skipjack. Впрочем, в случае анализа алгоритмов с подобными модификациями запас криптостойкости можно рассматривать только как качественную характеристику, имеющую достаточно косвенное отношение к исследуемому алгоритму шифрования.

1.6. Криптоаналитические методы, используемые в атаках

В предыдущем разделе были классифицированы атаки на алгоритмы симметричного шифрования. Теперь опишем существующие криптоаналитические методы, переходя от более простых к более сложным.

Метод «грубой силы»

Метод «грубой силы» (brute-force attack) предполагает перебор всех возможных вариантов ключа шифрования до нахождения искомого ключа.

Пусть размер ключа шифрования в битах равен b. Соответственно, существует 2^b вариантов ключа. Криптоаналитик должен перебрать все возможные ключи, т. е. (если рассматривать b-битную последовательность как число) применить в качестве ключа значение 0, затем 1, 2, 3 и т. д. до максимально возможного $(2^b - 1)$. В результате ключ шифрования обязательно будет найден, причем в среднем такой поиск потребует $2^b/2$, т. е. 2^{b-1} тестовых операций шифрования.

Понятно, что необходим какой-либо критерий корректности найденного ключа. С атакой с известным открытым текстом все достаточно просто — при тестировании каждого ключа K_x шифртекст C расшифровывается (в результате получается некое значение M') и сравнивается с соответствующим ему открытым текстом M; совпадение $M' = M$ говорит о том, что искомый ключ найден.

Несколько сложнее с атакой на основе шифртекста. В этом случае необходимо наличие какой-либо дополнительной информации об открытом тексте, например, следующей.

- Если открытый текст является осмысленным текстом на каком-либо языке, перехваченный шифртекст должен иметь достаточный размер для однозначного расшифровывания в осмысленный текст (минимально достаточ-

ный для этого размер называется *точкой единственности*). В основополагающей для современных симметричных криптосистем работе [26] точка единственности для английского языка теоретически определена как 27 букв. Если сообщение короче, то при переборе возможно появление нескольких различных осмысленных текстов, каждому из которых соответствует некий кандидат в искомые ключи. При невозможности перехвата дополнительных шифртекстов невозможно определить, какое из осмысленных сообщений является верным, если это не ясно из контекста.

- Если открытый текст состоит из бинарных данных, необходима какая-либо информация о том, что он из себя представляет. Если перехватывается архив, то при переборе ключей каждое значение M' должно рассматриваться как возможный заголовок архива. При другом потенциальном M это может быть PE-заголовок исполняемого файла для Windows, заголовок графического файла и т. д.
- Стоит отметить, что многие средства шифрования информации внедряют в формат зашифрованного объекта контрольную сумму открытого текста для проверки его целостности после расшифровывания (см. подробное описание в [14]). Это может быть, например, имитовставка, вычисленная согласно отечественному криптостандарту ГОСТ 28147-89 (*см. разд. 3.1*), или просто CRC32. Главное, что такая контрольная сумма может быть идеальным эталоном при криптоанализе, вполне подходящим для определения верного ключа.

Защита от атак, выполняемых методом «грубой силы», весьма проста — достаточно лишь увеличить размер ключа. Понятно, что увеличение размера ключа на 1 бит увеличит возможное количество ключей (и среднее время атаки) в 2 раза.

Несмотря на простоту атаки методом «грубой силы», существуют различные методы улучшения ее эффективности, например, следующие.

- Атака методом «грубой силы» простейшим образом распараллеливается: при наличии, скажем, миллиона компьютеров, участвующих в атаке, ключевое множество делится на миллион равных фрагментов, которые распределяются между участниками атаки [95].
- Скорость перебора ключей может быть во много раз увеличена, если в переборе участвуют не компьютеры общего назначения, а специализированные устройства. В [95] приводится пример микросхемы ORCA компании AT&T (технология ПЛИС), которая способна перебрать до 30 миллионов ключей DES (подробную информацию *см. в разд. 3.15*) в секунду. В той же работе [95] рассмотрено применение для перебора ключей специализированных микросхем, каждая из которых способна перебрать

до 200 миллионов ключей DES в секунду. Следует учесть, что приведенные в [95] оценки даны на конец 1995 г. — сейчас перебор может осуществляться несравнимо быстрее.

Все эти методы были опробованы на американском стандарте шифрования DES.

Понятно, что с развитием вычислительной техники требования к размеру ключа шифрования постоянно возрастают. В той же работе [95] был рекомендован 90-битный размер ключа в качестве абсолютно безопасного (причем с 20-летним запасом) на конец 1995 г. Сейчас подавляющее большинство алгоритмов шифрования используют ключи размером от 128 битов, что считается безопасным с примерно 80-летним запасом [131].

Современная вычислительная техника не позволяет «в лоб» атаковать 128-битный ключ полным перебором. Однако атаки методом «грубой силы» часто используются в контексте других атак — например, с помощью дифференциального криптоанализа (этот метод будет описан далее) сужается область возможных ключей, после чего выполняется перебор оставшихся вариантов (один из вариантов подобной атаки описан, например, в работе [59]).

Атаки класса «встреча посередине»

Как было сказано выше, алгоритм шифрования считается стойким, если атака методом «грубой силы» против него неэффективна, а более быстрых возможностей вскрыть алгоритм не существует [95].

Любые методы, способные вскрыть алгоритм шифрования быстрее, чем полный перебор ключей, эксплуатируют те или иные конструктивные недостатки алгоритма или его реализации. Начнем обзор таких методов с атак класса *встреча посередине* (meet-in-the-middle).

Простейший пример подобной атаки — вскрытие любого алгоритма шифрования, представляющего собой двойное шифрование данных с помощью какого-либо «одинарного» алгоритма. Рассмотрим вкратце алгоритм Double DES (подробную информацию *см. в разд. 3.15*), представляющий собой двойное шифрование обычным алгоритмом DES (рис. 1.13):

$$C = DES_{k_{2/2}}(DES_{k_{1/2}}(M)),$$

где $k_{1/2}$ и $k_{2/2}$ — половины двойного ключа алгоритма Double DES, каждая из которых представляет собой обычный 56-битный ключ DES.

Double DES решает основную проблему алгоритма DES (56-битный ключ шифрования слишком короток, как было показано выше) — 112-битный ключ Double DES невозможно вскрыть полным перебором. Однако вскрытие

Double DES легко выполняется атакой на основе известных открытых текстов. Предположим, у криптоаналитика есть открытый текст $M1$ и результат его зашифровывания — $C1$. Он может выполнить следующую последовательность действий (рис. 1.14):

1. Выполняется зашифровывание $DES_{kx}(M1)$ на всем ключевом множестве $(kx = 0...2^{56} - 1)$ с записью результатов в некоторую таблицу.
2. Производится расшифровывание $DES^{-1}{}_{ky}(C1)$ также на всем ключевом множестве; результаты расшифровывания сравниваются со всеми записями в таблице, сформированной на шаге 1.
3. Если какой-либо результат, полученный на шаге 2, совпал с одним из результатов шага 1, то можно предположить, что нужный ключ найден, т. е. найдены соответствующие совпадающему результату $kx = k_{1/2}$ и $ky = k_{2/2}$.

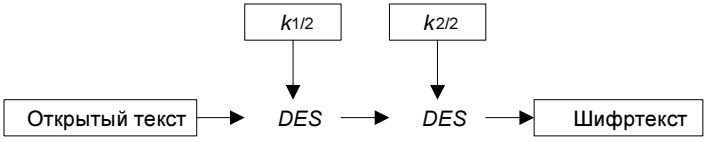


Рис. 1.13. Алгоритм Double DES

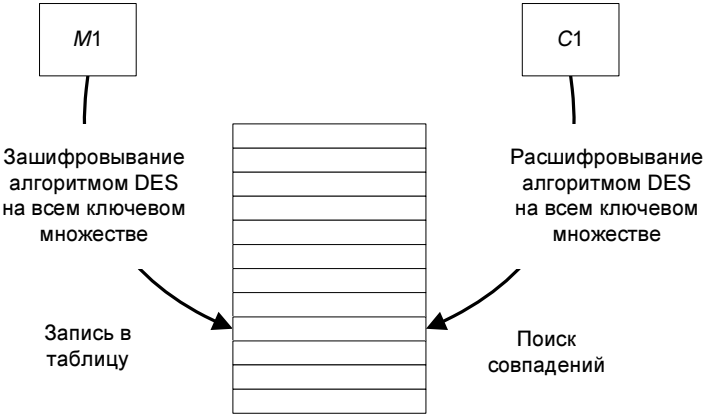


Рис. 1.14. Пример атаки «встреча посередине»

Следует учесть, что таких совпадений может быть много — порядка 2^{48} согласно [3]. Для «уточнения» правильного ключа из этих примерно 2^{48} вариантов достаточно наличия еще одной пары текстов: $M2$ и $C2$. Криптоаналитик может использовать их абсолютно так же, как и первую пару ($M1$, $C1$),

но перебор вариантов *kx* и *ky* осуществляется уже только по совпадениям первого перебора. В результате будет найден единственно верный ключ (вероятность повторного совпадения ничтожно мала — около 2^{-16} [3]; в этом случае используется третья пара — *M*3 и *C*3 — при ее наличии).

В результате воздействия данной атаки сложность вычисления ключа Double DES всего в 2 раза выше, чем полный перебор ключей обычного DES, что несравнимо меньше 2^{112} возможных вариантов «двойного» ключа. Алгоритм Double DES, собственно, и не используется, а упоминается лишь в иллюстрациях к атаке «встреча посередине» (см., например, [3] или [14]).

Атака «встреча посередине» была изобретена в 1981 г. известными криптологами Ральфом Мерклем (Ralph C. Merkle) и Мартином Хеллманом (Martin E. Hellman) именно в применении к алгоритму Double DES [266]. Кроме того, эта атака (но в заметно более сложном исполнении) применима и к одному из вариантов «тройного» DES (Triple DES) [266].

Впоследствии атаки данного класса были неоднократно использованы для анализа различных алгоритмов шифрования; наиболее показательные примеры относятся к следующим алгоритмам шифрования:

- атака «встреча посередине» позволяет вычислить 192-битный ключ алгоритма DEAL (*см. разд. 3.14*) выполнением порядка 2^{166} тестовых операций шифрования; для вычисления 256-битного ключа необходимо порядка 2^{222} операций [47]; и то, и другое не является практически осуществимым, но доказывает невысокий запас криптостойкости алгоритма DEAL;
- столь же непрактичная атака возможна против еще одного участника конкурса AES — SAFER+ (*см. разд. 3.45*) — для его вскрытия необходимо 2^{240} операций шифрования (при 256-битном ключе) [198].

Как видно, в отличие от Double DES, атака малоэффективна против более современных алгоритмов шифрования, что не удивительно. Однако, как и атака методом «грубой силы», атака «встреча посередине» нередко применима в комбинации с другими атаками, а также для оценки запаса криптостойкости модифицированных версий алгоритмов и алгоритмов с усеченным количеством раундов.

Дифференциальный криптоанализ

Этот метод атак на алгоритмы шифрования был изобретен в 1990 г. известными израильскими криптологами Эли Бихамом (Eli Biham) и Эди Шамиром (Adi Shamir) и опубликован в их работе [75]. Однако не менее известный криптолог Брюс Шнайер в своей книге [28] утверждает, что дифференциальный

криптоанализ был открыт существенно раньше, но не появлялся в открытой печати. Тем не менее, именно Бихам и Шамир до сих пор считаются изобретателями дифференциального криптоанализа.

Работа [75] посвящена атаке с помощью дифференциального криптоанализа на алгоритм DES (*см. разд. 3.15*). Кратко опишем функцию раунда алгоритма DES $f(b,k)$ (рис. 1.15):

1. Над 32-битным входным значением b выполняется расширяющая перестановка (EP), которая дает 48-битное значение be.

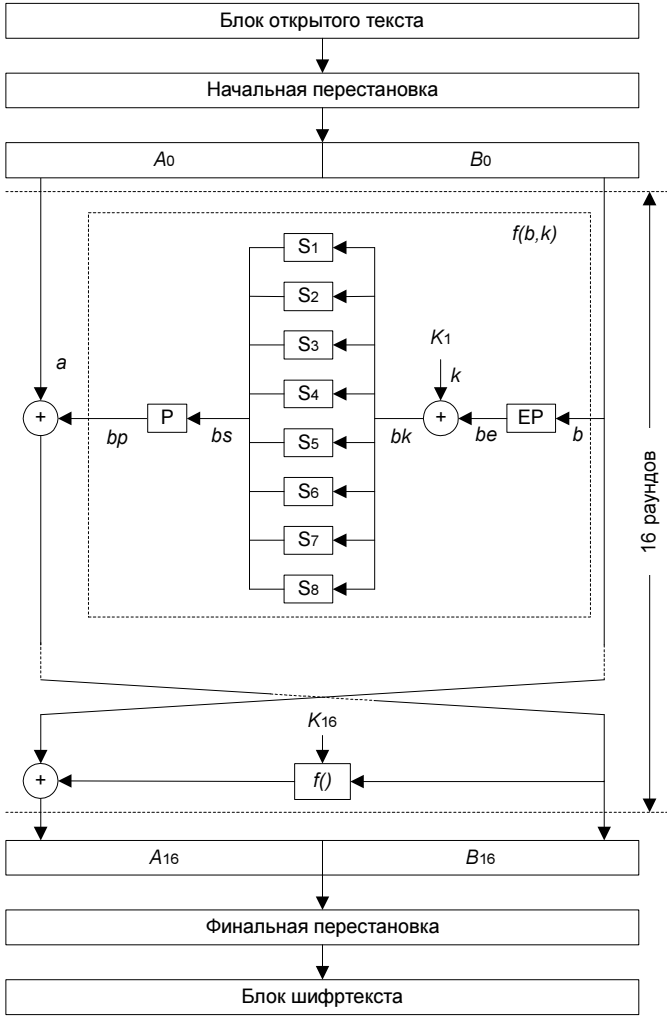


Рис. 1.15. Структура алгоритма DES

2. *be* складывается с ключом раунда K_i операцией XOR (обозначим результат этой операции как *bk*).
3. *bk* разбивается на 8 фрагментов по 6 битов, каждый из которых прогоняется через соответствующую таблицу замен (S_1 … S_8). Каждая таблица включает в себя 4 строки, содержащие по 16 значений от 0 до 15. Входное значение интерпретируется следующим образом: два крайних бита формируют номер строки (от 0 до 3), из которой выбирается число, расположенное в столбце, номер которого соответствует значению четырех остальных битов входа. Например, при двоичном входе 101100 (десятичное число 44) выбирается значение шестой ячейки второй строки.
4. Полученные в результате замен 4-битные значения объединяются в 32-битный субблок (обозначим его как *bs*), после чего над ними выполняется еще одна перестановка (*P*), которая завершает работу функции *f*.

Теперь рассмотрим собственно дифференциальный криптоанализ. Этот метод основан на анализе пар открытых текстов, между которыми существует определенная *разность* (difference). При анализе алгоритмов разность текстов может быть определена различным образом. Для DES разность открытых текстов M_1 и M_2 определяется как операция XOR между данными текстами:

$$\Delta = M_1 \oplus M_2.$$

Дифференциальный криптоанализ использует множество пар текстов с определенной разностью, анализ которых позволяет выделить некий ключ (или его фрагмент), который является искомым ключом либо однозначно, либо с наибольшей (по сравнению с другими возможными ключами) вероятностью.

Выполняется такой анализ следующим образом. Предположим, имеется два открытых текста, которые на входе в функцию *f* какого-либо раунда алгоритма имеют разность Δ_b (рис. 1.15):

$$\Delta_b = b_1 \oplus b_2.$$

Разность $\Delta_{be} = be_1 \oplus be_2$, т. е. разность после обработки b_1 и b_2 расширяющей перестановкой *EP*, весьма легко определить, поскольку:

$$\Delta_{be} = EP(b_1) \oplus EP(b_2) = EP(b_1 \oplus b_2) = EP(\Delta_b).$$

Наложение фрагмента ключа операцией XOR вообще не меняет разность, т. е.:

$$\Delta_{bk} = \Delta_{be}.$$

Аналогично преобразованию *EP*, легко вычислить разность Δ_{bp} после перестановки *P*:

$$\Delta_{bp} = P(bs_1) \oplus P(bs_2) = P(bs_1 \oplus bs_2) = P(\Delta_{bs}).$$

Таким образом, единственной операцией из выполняемых функцией *f*, существенно влияющей на значение разности, остается табличная замена. Значение Δ_{bs} зависит не только от разности Δ_{bk}, но и от конкретных входных значений bk_1 и bk_2. Здесь-то и проявляется влияние наложенного предыдущей операцией ключа шифрования.

Как было сказано выше, таблицы замен меняют 6-битное входное значение на 4-битное. Это означает, что любой входной разности $\Delta_{bk,n}$ (где n — номер таблицы) соответствует $2^6 = 64$ возможных пар входных значений (обозначим их $bk_{1,n}$ и $bk_{2,n}$). Соответственно, любой выходной разности $\Delta_{bs,n}$ соответствует $2^4 = 16$ возможных пар $bs_{1,n}$ и $bs_{2,n}$. Дифференциальный криптоанализ алгоритма DES эксплуатирует тот факт, что, во-первых, каждое конкретное значение $\Delta_{bk,n}$ приводит не ко всем возможным 16 значениям $\Delta_{bs,n}$, а во-вторых, данные значения $\Delta_{bs,n}$ имеют весьма различную вероятность. Бихам и Шамир в качестве примера рассматривают таблицу S_1 и ее входную разность $\Delta_{bk,1} = 34$ (здесь и далее значения разностей указываются в шестнадцатеричном виде). Возможные значения $\Delta_{bs,1}$ и их вероятности (в количестве значений пар $bk_{1,1}$ и $bk_{2,1}$ из 64 возможных, которые приводят к данному значению $\Delta_{bs,1}$) приведены в табл. 1.1 [75].

Таблица 1.1

Значение $\Delta_{bs,1}$	**Вероятность**
1	8
2	16
3	6
4	2
7	12
8	6
D	8
F	6
Остальные	0
Всего	64

Как с помощью всего этого можно определить ключ, рассмотрим на простейшем примере. Предположим, что в распоряжении криптоаналитика имеется пара текстов с $\Delta_{bk,1} = 34$. Кроме того, у криптоаналитика есть соответ-

ствующие им шифртексты (снова предположим, что для их зашифровывания использовался *однораундовый* DES) с $\Delta_{bs,1} = 4$. Входные значения $S1$ при таких условиях известны [75] — это значения 13 и 27 ($bk_{1,1} = 13$, а $bk_{2,1} = 27$, или наоборот; здесь также указаны шестнадцатеричные значения). Получается, что криптоаналитик знает значения be_1 и be_2 (поскольку ему известны открытые тексты), а также два варианта значений $bk_{1,1}$ и $bk_{2,1}$. В результате криптоаналитик может простейшей операцией XOR вычислить 2 возможных варианта первых шести битов K_1. Выбрать из двух вариантов правильный криптоаналитику поможет вторая пара открытых текстов, если таковая у него есть. Даже если такой пары нет, криптоаналитик существенно сузил область возможных значений ключа: в 2^5 раз.

Понятно, что однораундовый DES не является реальным объектом для криптоанализа. Для вскрытия алгоритмов с бóльшим количеством раундов используют *характеристики* — такие разности открытых текстов, которые с высокой вероятностью вызывают определенные разности в получаемых шифртекстах [28]. В качестве примера рассмотрим следующую трехраундовую характеристику алгоритма DES (рис. 1.16) [75]:

1. В первом раунде на вход функции *f* подаются два субблока с разностью Δ_b = 60 00 00 00. Разность подобрана таким образом, чтобы после перестановки *EP* (наложение фрагмента ключа, как было сказано выше, не влияет на разность) она была нулевой на входе всех таблиц замен, кроме S_1, на входе которой разность $\Delta_{bk,1} = 0\text{C}$. Эта разность обеспечивает выходную разность $\Delta_{bs,1} = 0\text{E}$ с вероятностью $14/64$, которая после перестановки *P* (с учетом нулевой разности на выходе остальных таблиц) приводит к разности Δ_{bp} = 00 80 82 00. Как видно, эта разность совпадает с разностью необработанных субблоков, т. е. Δ_a; в результате наложения результата функции *f* на необработанный субблок получается нулевая разность на входе второго раунда.
2. Нулевая разность Δ_b на входе функции *f* во втором раунде дает также нулевую разность на выходе. Следовательно (рис. 1.16), разность на входе функции *f* третьего раунда полностью эквивалентна таковой в первом раунде.
3. Получается, что разности третьего раунда и их вероятности аналогичны первому раунду. Таким образом, данная характеристика обеспечивает после трех раундов выходную разность Δ = 00 80 82 00 60 00 00 00 (равную входной разности) со следующей вероятностью:

$$p = \frac{14}{64} * 1 * \frac{14}{64} = \left(\frac{14}{64}\right)^2 \approx 0,048.$$

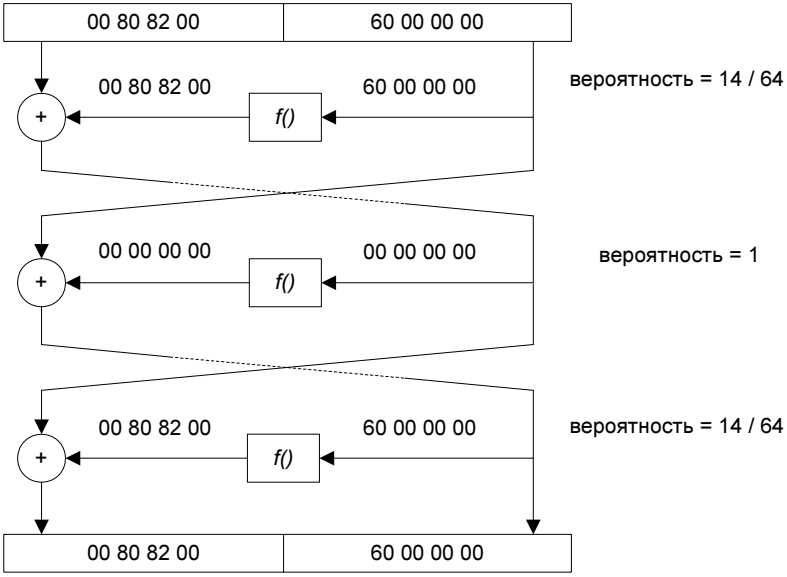


Рис. 1.16. Пример трехраундовой характеристики

Характеристики используются следующим образом (на примере *трехраундового* DES и приведенной выше характеристики):

1. Генерируется необходимое количество выбранных открытых текстов (пары которых соответствуют выбранной характеристике) и соответствующих им шифртекстов. Количество пар с требуемой разностью оценивается в [75] как «некоторое число» (several), умноженное на p^{-1}.
2. Формируется таблица, содержащая возможные варианты искомого фрагмента ключа шифрования (как было показано выше для одноraундового DES). Верное значение должно повторяться для каждого анализируемого текста, поскольку именно оно использовалось для его зашифровывания.

Версии алгоритма с бóльшим количеством раундов (включая полную 16-раундовую) вскрываются различными путями (см., например, [75] и [78]).

- Использованием характеристик, распространяющихся на большее количество раундов (но с меньшей вероятностью). Например, в [75] приведена 5-раундовая характеристика для DES с вероятностью 0,000095.
- Параллельное использование двух и более различных характеристик.
- Использование итеративных характеристик. Пример итеративной характеристики для DES приведен на рис. 1.17. Итеративные характеристики могут «размножаться» на требуемое количество раундов (конечно, также

с потерей в вероятности), поскольку их структура подразумевает, что в выходной разности значения Δ_a и Δ_b меняются местами относительно входной разности (что необходимо для «размножения» характеристики, поскольку субблоки в DES меняются местами в конце раунда).

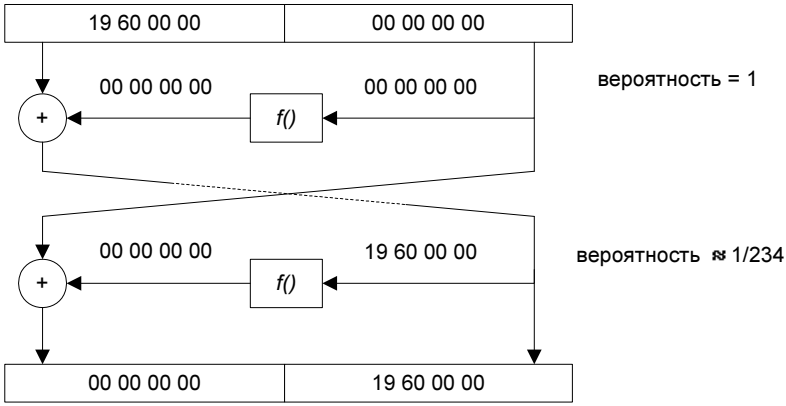


Рис. 1.17. Пример итеративной характеристики

Поиск характеристик с достаточной вероятностью является творческим процессом. На данный момент автору не известны какие-либо методы, позволяющие алгоритмизировать поиск характеристик для различных алгоритмов шифрования с высокой вероятностью. Стоит сказать и о том, что современные алгоритмы обладают весьма различной криптостойкостью к дифференциальному криптоанализу.

Если дифференциальный криптоанализ и был известен еще в середине 1970-х гг. (см. выше), то явно не всем разработчикам алгоритмов шифрования — многие из более поздних алгоритмов могут быть вскрыты этим методом. Рассмотрим наиболее показательные примеры.

- Известный алгоритм RC2 (*см. разд. 3.41*) вскрывается дифференциальным криптоанализом при наличии 2^{59} выбранных открытых текстов [224].
- Не менее известный и широко используемый алгоритм RC5 (а именно, его основной вариант — RC5-32/12/16 — *см. разд. 3.42*) еще менее стоек к дифференциальному криптоанализу — он вскрывается при наличии 2^{44} выбранных открытых текстов [90]. Существуют и другие варианты алгоритма RC5, также подверженные дифференциальному криптоанализу — подробную информацию *см. в разд. 3.42*.

- «Ускоренный» вариант алгоритма ICE — Thin-ICE (*см. разд. 3.24*) — вскрывается с вероятностью 25 % при наличии 2^{23} выбранных открытых текстов; при увеличении количества текстов до 2^{27} вероятность вскрытия алгоритма возрастает до 95 % [377].
- Один из вариантов алгоритма DES — Generalized DES (GDES) с 16 раундами и размером блока 256 битов — вскрывается при наличии всего 6 (!) выбранных открытых текстов [75]. Некоторые другие варианты алгоритма DES (DESX, DES с независимыми подключами, s^2DES и др.) также подвержены дифференциальному криптоанализу — подробную информацию *см. в разд. 3.15*.

Кроме того, известно множество работ, посвященных вскрытию дифференциальным криптоанализом версий различных алгоритмов с усеченным количеством раундов.

Линейный криптоанализ

Линейный криптоанализ изобрел японский криптолог Мицуру Мацуи (Mitsuru Matsui). Предложенный им в 1993 г. метод изначально был направлен на вскрытие того же алгоритма DES [256]. Впоследствии линейный криптоанализ был распространен и на другие алгоритмы [326]. Однако есть источники (например, [395]), которые утверждают, что линейный криптоанализ был изобретен для вскрытия алгоритма FEAL (*см. разд. 3.18*) в 1992 г., а уже затем перенесен на DES.

Смысл линейного криптоанализа состоит в нахождении соотношений следующего вида [28, 256]:

$$P_{i1} \oplus P_{i2} \oplus ... \oplus P_{ia} \oplus C_{j1} \oplus C_{j2} \oplus ... \oplus C_{jb} = K_{k1} \oplus K_{k2} \oplus ... \oplus K_{kc}, \quad (1.1)$$

где P_n, C_n и K_n — n-е биты открытого текста, шифртекста и ключа соответственно.

Для произвольно выбранных битов открытого текста, шифртекста и ключа вероятность P справедливости такого соотношения составляет около $1/2$. В том случае, если криптоаналитику удается найти такие биты, при которых вероятность P заметно отличается от $1/2$, данным соотношением можно воспользоваться для вскрытия алгоритма.

Так же, как и в дифференциальном криптоанализе, сначала криптоаналитик находит некое одноraундовое соотношение, затем пытается распространить его на большее количество раундов, в результате находит соотношение для полнораундового варианта анализируемого алгоритма. Стоит отметить, что,

в отличие от дифференциального криптоанализа, существуют алгоритмы поиска полезных соотношений; два подобных алгоритма изначально были описаны Мицуру Мацуи в [256], другие появились позже (см., например, [180]).

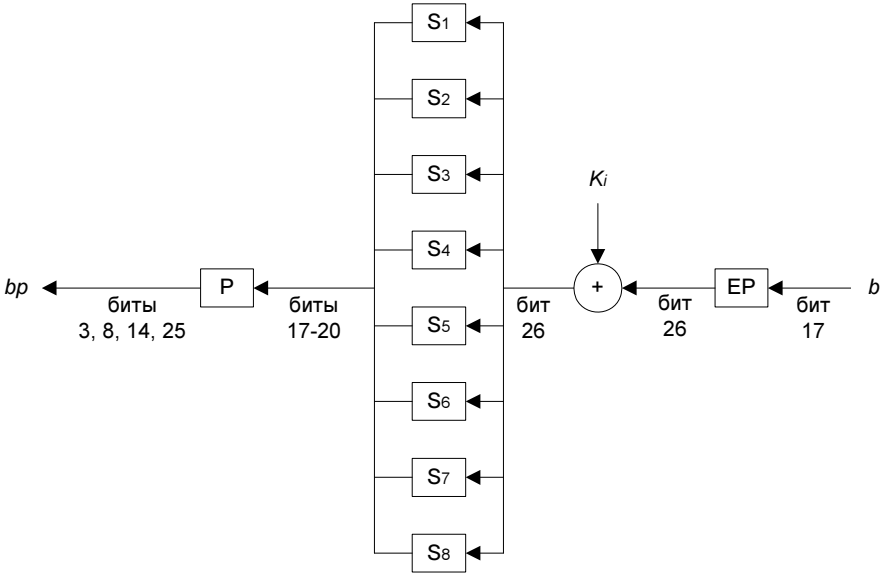


Рис. 1.18. Наиболее эффективное однораундовое соотношение для алгоритма DES

На рис. 1.18 представлено наиболее эффективное однораундовое соотношение для алгоритма DES [28, 256]. Данное соотношение использует то свойство таблицы S_5, что второй входной бит таблицы равен результату применения операции XOR над всеми четырьмя выходными битами с вероятностью $3/16$ (т. е. смещение в $5/16$ относительно вероятности $1/2$), т. е. (применяя нотацию, использованную ранее при описании дифференциального криптоанализа):

$$(bk_5)_2 = (bs_5)_1 \oplus (bs_5)_2 \oplus (bs_5)_3 \oplus (bs_5)_4,$$

что эквивалентно

$$bk_{26} = bs_{17} \oplus bs_{18} \oplus bs_{19} \oplus bs_{20}. \qquad (1.2)$$

Шнайер в [28] пишет, что данное свойство таблицы S_5 заметил еще Эди Шамир в 1985 г., однако именно Мацуи удалось его использовать для атаки на алгоритм. Дальнейшее расширение данного свойства таково:

1. Как показано на рис. 1.18, соотношение (1.2) распространяется на целый раунд алгоритма; в результате чего (с учетом перестановок) получается следующее однораундовое соотношение:

$$b_{17} \oplus bp_3 \oplus bp_8 \oplus bp_{14} \oplus bp_{25} = (K_i)_{26},$$

где i — номер раунда.

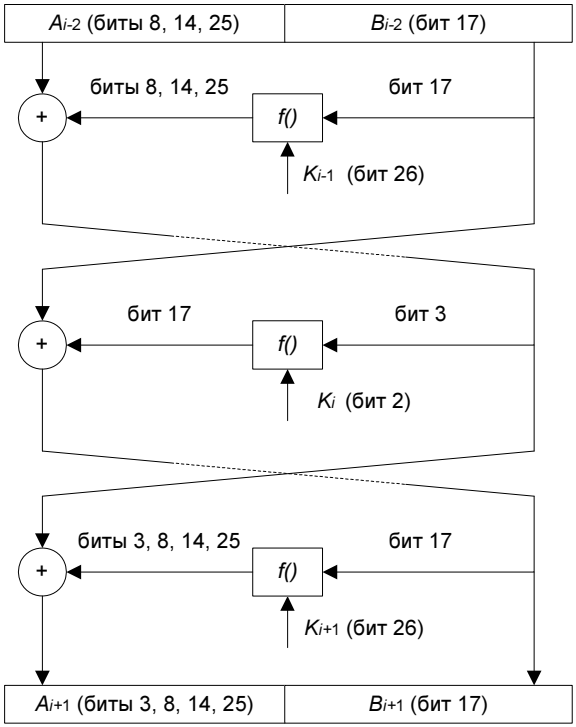


Рис. 1.19. Пример трехраундового соотношения

2. Весьма похоже на дифференциальный криптоанализ соотношение распространяется на несколько раундов. Пример такого соотношения показан на рис. 1.19 (соотношение (1.1) справедливо для тех битов, номера которых указаны на рисунке); его вероятность смещена относительно $1/2$ на 0,0061 [28]. А для полнораундового DES известно соотношение, выполняющееся с вероятностью $1/2 + 2^{-24}$ [86].

3. С использованием максимально эффективного соотношения выполняется анализ имеющихся пар «открытый текст — шифртекст» с целью найти наиболее вероятные значения определенных битов ключа шифрования.

Весьма часто линейный криптоанализ используется в совокупности с атакой методом «грубой силы» — определенные биты ключа находятся с помощью линейного криптоанализа, после чего выполняется исчерпывающий поиск по возможным значениям остальных битов. Подобным образом алгоритм DES вскрывается при наличии 2^{43} известных открытых текстов, что существенно эффективнее дифференциального криптоанализа [28, 263].

Стоит отметить и то, что для дифференциального криптоанализа требуются обычно выбранные открытые тексты, тогда как метод линейного криптоанализа «довольствуется» известными открытыми текстами, что существенно увеличивает область применения этого метода. Однако, если это возможно, и в линейном криптоанализе в ряде случаев бывает весьма полезно использовать выбранные открытые тексты вместо известных. В частности, для DES существует методика, позволяющая существенно уменьшить количество требуемых данных и вычислений использованием выбранных открытых текстов [366].

Линейный криптоанализ имеет одно весьма полезнейшее свойство: при определенных обстоятельствах соотношение (1.1) может быть преобразовано к следующему:

$$C_{j1} \oplus C_{j2} \oplus ... \oplus C_{jb} = K_{k1} \oplus K_{k2} \oplus ... \oplus K_{kc}.$$

В этом соотношении полностью отсутствуют биты открытого текста, т. е. с помощью линейного криптоанализа можно построить атаку на основе только шифртекста (*см. разд. 1.5*), что еще больше расширяет область применения линейного криптоанализа, поскольку атака, для которой требуется только перехваченный шифртекст, является наиболее практичной. В [256] Мацуи описывает такую атаку на полнораундовый DES, которая срабатывает при допущении, что соответствующий перехваченным шифртекстам открытый текст представляет собой текст на английском языке в ASCII-кодировке. Для этой атаки криптоаналитику потребуется порядка 2^{54} шифртекстов.

Помимо DES и упомянутого выше алгоритма FEAL, известны и другие алгоритмы, подверженные линейному криптоанализу, например (отметим, что линейному криптоанализу подвержено существенно меньше известных алгоритмов, чем дифференциальному):

- ❒ линейный криптоанализ действует против алгоритма RC5 (*см. разд. 3.42*) в случае, если искомый ключ шифрования попадает в класс слабых ключей [179];