Лекція 10. Шифрування. Модель традиційного шифрування; криптографія, криптоаналіз.

В контексте сетевых коммуникаций можно выделить следующие типы нарушений защиты.

1. **Раскрытие содержимого.** Открытие содержимого сообщения любому лицу или процессу, не владеющему соответствующим криптографическим ключом.
2. **Анализ потока данных.** Раскрытие структуры потока обмена данными участвующих в таком обмене сторон. В случае ориентированного на установление логических соединений приложения может быть определена частота и длительность соединений. И в среде, ориентированной на установление логических соединений, и при отсутствии таковых могут быть определены число и длина сообщений, пересылаемых участвующими в обмене сторонами.
3. **Имитация.** Внедрение в поток сообщений от ложного источника. Здесь подразумевается и создание сообщений противником, выдаваемых им за сообщения от уполномоченного объекта, и обманные подтверждения получения или неполучения сообщений лицом, не являющимся истинным адресатом сообщения.
4. **Модификация содержимого.** Изменение содержимого сообщения, включая вставку, удаление, реорганизацию или любую другую модификацию сообщения.
5. **Модификация последовательности сообщений.** Любая модификация последовательности сообщений, формирующих поток обмена данными, включая вставку, удаление или изменение порядка следования сообщений.
6. **Модификация временных характеристик.** Задержка или воспроизведение сообщений. В случае ориентированного на установление логических соединений приложения весь сеанс или некоторая последовательность сообщений может быть воспроизведением некоторого ранее осуществленного реального сеанса или некоторой последовательности ранее переданных сообщений с определенными задержками во времени или повторениями. В приложении без установления соединений можно задержать или воспроизвести отдельное сообщение (например, дейтаграмму).
7. **Отречение.** Отрицание получения сообщения адресатом или отрицание факта передачи сообщения источником.

Меры, принимаемые в связи с первыми двумя типами атаки, относятся к конфиденциальности. Меры, принимаемые в связи с нарушениями, указанными в пунктах 3–6 приведенного выше списка, относятся к вопросам аутентификации сообщений. Механизмы решения проблем, возникающих в связи с атаками, представленными в п. 7, относятся к проблеме использования цифровых подписей, но, вообще, средства цифровой подписи учитывают многие вопросы, вытекающие из рассмотрения пунктов 3–6.

Криптография

Основные понятия

Рассмотрим общую схему симметричной, или традиционной, криптографии.

Рис. 2.1. Общая схема симметричного шифрования

В процессе шифрования используется определенный алгоритм шифрования, на вход которому подаются исходное незашифрованное сообщение, называемое также plaintext, и ключ. Выходом алгоритма является зашифрованное сообщение, называемое также ciphertext. Ключ является значением, не зависящим от шифруемого сообщения. Изменение ключа должно приводить к изменению зашифрованного сообщения.

Зашифрованное сообщение передается получателю. Получатель преобразует зашифрованное сообщение в исходное незашифрованное сообщение с помощью алгоритма дешифрования и того же самого ключа, который использовался при шифровании, или ключа, легко получаемого из ключа шифрования.

Незашифрованное сообщение будем обозначать P или M, от слов plaintext и message. Зашифрованное сообщение будем обозначать C, от слова ciphertext.

Безопасность, обеспечиваемая традиционной криптографией, зависит от нескольких факторов.

Во-первых, криптографический алгоритм должен быть достаточно сильным, чтобы передаваемое зашифрованное сообщение невозможно было расшифровать без ключа, используя только различные статистические закономерности зашифрованного сообщения или какие-либо другие способы его анализа.

Во-вторых, безопасность передаваемого сообщения должна зависеть от секретности ключа, но не от секретности алгоритма. Алгоритм должен быть проанализирован специалистами, чтобы исключить наличие слабых мест, при которых плохо скрыта взаимосвязь между незашифрованным и зашифрованным сообщениями. К тому же при выполнении этого условия производители могут создавать дешевые аппаратные чипы и свободно распространяемые программы, реализующие данный алгоритм шифрования.

В-третьих, алгоритм должен быть таким, чтобы нельзя было узнать ключ, даже зная достаточно много пар (зашифрованное сообщение, незашифрованное сообщение), полученных при шифровании с использованием данного ключа.

Клод Шеннон ввел понятия диффузии и конфузии для описания стойкости алгоритма шифрования.

Диффузия – это рассеяние статистических особенностей незашифрованного текста в широком диапазоне статистических особенностей зашифрованного текста. Это достигается тем, что значение каждого элемента незашифрованного текста влияет на значения многих элементов зашифрованного текста или, что то же самое, любой элемент зашифрованного текста зависит от многих элементов незашифрованного текста.

Конфузия – это уничтожение статистической взаимосвязи между зашифрованным текстом и ключом.

Если X – это исходное сообщение и K – криптографический ключ, то зашифрованный передаваемый текст можно записать в виде

$Y = E_K[X]$.

Получатель, используя тот же ключ, расшифровывает сообщение

$X = D_K[Y]$

Противник, не имея доступа к K и X, должен попытаться узнать X, K или и то, и другое.

Алгоритмы симметричного шифрования различаются способом, которым обрабатывается исходный текст. Возможно шифрование блоками или шифрование потоком.

Блок текста рассматривается как неотрицательное целое число, либо как несколько независимых неотрицательных целых чисел. Длина блока всегда выбирается равной степени двойки. В большинстве блочных алгоритмов симметричного шифрования используются следующие типы операций:

- Табличная подстановка, при которой группа битов отображается в другую группу битов. Это так называемые S-box.
- Перемещение, с помощью которого биты сообщения переупорядочиваются.
- Операция сложения по модулю 2, обозначаемая XOR или $\oplus$.
- Операция сложения по модулю 2^{32} или по модулю 2^{16}.
- Циклический сдвиг на некоторое число битов.

Эти операции циклически повторяются в алгоритме, образуя так называемые раунды. Входом каждого раунда является выход предыдущего раунда и ключ, который получен по определенному алгоритму из ключа шифрования K. Ключ раунда называется подключом. Каждый алгоритм шифрования может быть представлен следующим образом:

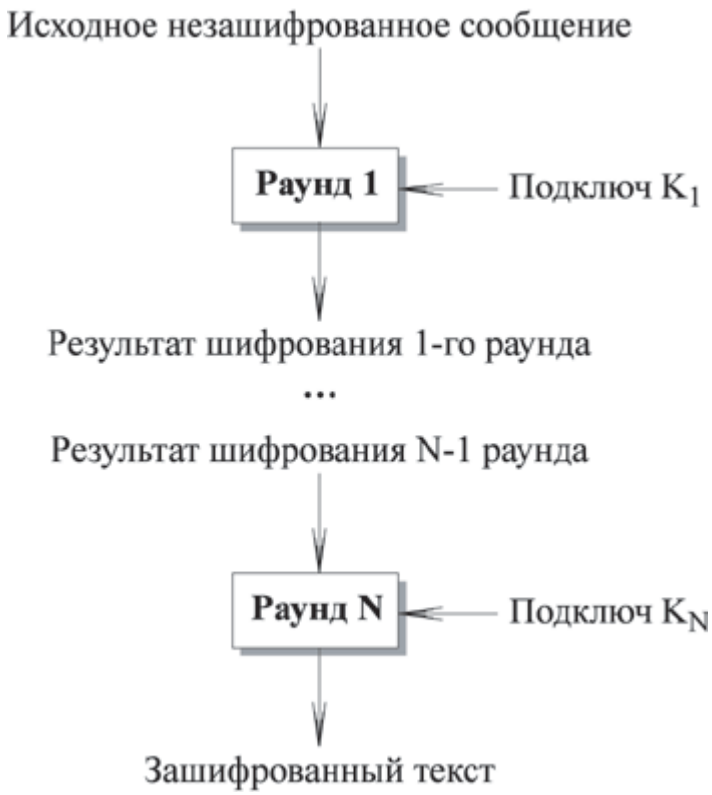

Рис. 2.2. Структура алгоритма симметричного шифрования

Области применения

Стандартный алгоритм шифрования должен быть применим во многих приложениях:

- Шифрование данных. Алгоритм должен быть эффективен при шифровании файлов данных или большого потока данных.
- Создание случайных чисел. Алгоритм должен быть эффективен при создании определенного количества случайных битов.
- Хэширование. Алгоритм должен эффективно преобразовываться в одностороннюю хэш-функцию.

Платформы

Стандартный алгоритм шифрования должен быть реализован на различных платформах, которые, соответственно, предъявляют различные требования.

- Алгоритм должен эффективно реализовываться на специализированной аппаратуре, предназначенной для выполнения шифрования/дешифрования.
- Большие процессоры. Хотя для наиболее быстрых приложений всегда используется специальная аппаратура, программные реализации применяются чаще. Алгоритм должен допускать эффективную программную реализацию на 32-битных процессорах.
- Процессоры среднего размера. Алгоритм должен работать на микроконтроллерах и других процессорах среднего размера.
- Малые процессоры. Должна существовать возможность реализации алгоритма на смарт-картах, пусть даже с учетом жестких ограничений на используемую память.

Дополнительные требования

Алгоритм шифрования должен, по возможности, удовлетворять некоторым дополнительным требованиям.

- Алгоритм должен быть простым для написания кода, чтобы минимизировать вероятность программных ошибок.
- Алгоритм должен иметь плоское пространство ключей и допускать любую случайную строку битов нужной длины в качестве возможного ключа. Наличие слабых ключей нежелательно.
- Алгоритм должен легко модифицироваться для различных уровней безопасности и удовлетворять как минимальным, так и максимальным требованиям.
- Все операции с данными должны осуществляться над блоками, кратными байту или 32-битному слову.

Сеть Фейштеля

Блочный алгоритм преобразовывает n-битный блок незашифрованного текста в n-битный блок зашифрованного текста. Число блоков длины n равно 2^n. Для того чтобы преобразование было обратимым, каждый из таких блоков должен преобразовываться в свой уникальный блок зашифрованного текста. При маленькой длине блока такая подстановка плохо скрывает статистические особенности незашифрованного текста. Если блок имеет длину 64 бита, то он уже хорошо скрывает статистические особенности исходного текста. Но в данном случае преобразование текста не может быть произвольным в силу того, что ключом будет являться само преобразование, что исключает эффективную как программную, так и аппаратную реализации.

Наиболее широкое распространение получили сети Фейштеля, так как, с одной стороны, они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой стороны, достаточно просты и компактны.

Сеть Фейштеля имеет следующую структуру. Входной блок делится на несколько равной длины подблоков, называемых ветвями. В случае, если блок имеет длину 64 бита, используются две ветви по 32 бита каждая. Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево. Такое преобразование выполняется несколько циклов или раундов. В случае двух ветвей каждый раунд имеет структуру, показанную на рисунке:

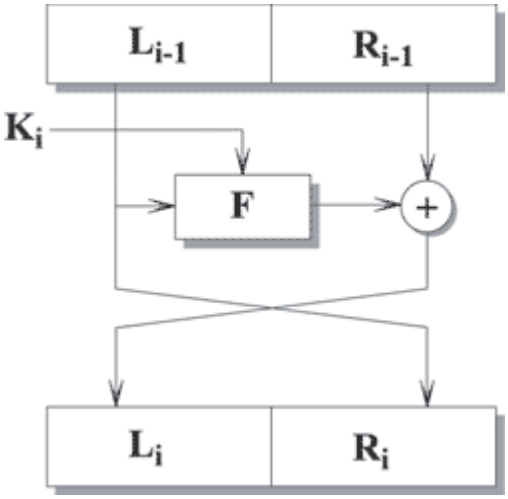

Рис. 2.3. I-ый раунд сети Фейштеля

Функция F называется образующей. Каждый раунд состоит из вычисления функции F для одной ветви и побитового выполнения операции XOR результата F с другой ветвью. После этого ветви меняются местами. Считается, что оптимальное число раундов - от 8 до 32. Важно то, что увеличение количества раундов значительно увеличивает криптостойкость алгоритма. Возможно, эта особенность и повлияла на столь активное распространение сети Фейштеля, так как для большей криптостойкости достаточно просто увеличить количество раундов, не изменяя сам алгоритм. В последнее время количество раундов не фиксируется, а лишь указываются допустимые пределы.

Сеть Фейштеля является обратимой даже в том случае, если функция F не является таковой, так как для дешифрования не требуется вычислять F^{-1}. Для дешифрования используется тот же алгоритм, но на вход подается зашифрованный текст, и ключи используются в обратном порядке.

В настоящее время все чаще используются различные разновидности сети Фейштеля для 128-битного блока с четырьмя ветвями. Увеличение количества ветвей, а не размерности каждой ветви связано с тем, что наиболее популярными до сих пор остаются процессоры с 32-разрядными словами, следовательно, оперировать 32-разрядными словами эффективнее, чем с 64-разрядными.

Основной характеристикой алгоритма, построенного на основе сети Фейштеля, является функция F. Различные варианты касаются также начального и конечного преобразований. Подобные преобразования, называемые забеливанием (whitening), осуществляются для того, чтобы выполнить начальную рандомизацию входного текста.

Криптоанализ

Процесс, при котором предпринимается попытка узнать X, K или и то, и другое, называется криптоанализом. Одной из возможных атак на алгоритм шифрования является лобовая атака, т.е. простой перебор всех возможных ключей. Если множество ключей достаточно большое, то подобрать ключ нереально. При длине ключа n бит количество возможных ключей равно 2^n. Таким образом, чем длиннее ключ, тем более стойким считается алгоритм для лобовой атаки.

Существуют различные типы атак, основанные на том, что противнику известно определенное количество пар незашифрованное сообщение – зашифрованное сообщение. При анализе зашифрованного текста противник часто применяет статистические методы анализа текста. При этом он может иметь общее представление о типе текста, например, английский или русский текст, выполнимый файл конкретной ОС, исходный текст на некотором конкретном языке программирования и т.д. Во многих случаях криптоаналитик имеет достаточно много информации об исходном тексте. Криптоаналитик может иметь возможность перехвата одного или нескольких незашифрованных сообщений вместе с их зашифрованным видом. Или криптоаналитик может знать основной формат или основные характеристики сообщения. Говорят, что криптографическая схема абсолютно безопасна, если зашифрованное сообщение не содержит никакой информации об исходном сообщении. Говорят, что криптографическая схема вычислительно безопасна, если:

1. Цена расшифровки сообщения больше цены самого сообщения.
2. Время, необходимое для расшифровки сообщения, больше срока жизни сообщения.

Дифференциальный и линейный криптоанализ

Рассмотрим в общих чертах основной подход, используемый при дифференциальном и линейном криптоанализе. И в том, и в другом случае предполагается, что известно достаточно большое количество пар (незашифрованнный текст, зашифрованный текст).

Понятие дифференциального криптоанализа было введено Эли Бихамом (Biham) и Ади Шамиром (Shamir) в 1990 году. Конечная задача дифференциального криптоанализа – используя свойства алгоритма, в основном свойства S-box, определить подключ раунда. Конкретный способ дифференциального криптоанализа зависит от рассматриваемого алгоритма шифрования.

Если в основе алгоритма лежит сеть Фейштеля, то можно считать, что блок m состоит из двух половин - m_0 и m_1. Дифференциальный криптоанализ рассматривает отличия, которые происходят в каждой половине при шифровании. (Для алгоритма DES "отличия" определяются с помощью операции XOR, для других алгоритмов возможен иной способ). Выбирается пара незашифрованных текстов с фиксированным отличием. Затем анализируются отличия, получившиеся после шифрования одним раундом алгоритма, и определяются вероятности различных ключей. Если для многих пар входных значений, имеющих одно и то же отличие X, при использовании одного и того же подключа одинаковыми (Y) оказываются и отличия соответствующих выходных значений, то можно говорить, что X влечет Y с определенной вероятностью. Если эта вероятность близка к единице, то можно считать, что подключ раунда найден с данной вероятностью. Так как раунды алгоритма независимы, вероятности определения подключа каждого раунда следует перемножать. Как мы помним, считается, что результат шифрования данной пары известен. Результаты дифференциального криптоанализа используются как при разработке конкретных S-box, так и при определении оптимального числа раундов.

Другим способом криптоанализа является линейный криптоанализ, который использует линейные приближения преобразований, выполняемых алгоритмом шифрования. Данный метод позволяет найти ключ, имея достаточно большое число пар (незашифрованный текст, зашифрованный текст). Рассмотрим основные принципы, на которых базируется линейный криптоанализ. Обозначим

$P[1],\ldots P[n]$ – незашифрованный блок сообщения.

$C[1],\ldots C[n]$ – зашифрованный блок сообщения.

$K[1],\ldots K[m]$ – ключ.

$A[i, j,\ldots k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$

Целью линейного криптоанализа является поиск линейного уравнения вида

$$P[\alpha_1, \alpha_2, \ldots, \alpha_a] \oplus C[\beta_1, \beta_2, \ldots, \beta_b] = K[\gamma_1, \gamma_2, \ldots \gamma_c]$$

Выполняющееся с вероятностью $p \neq 0.5$. α_i, β_i и γ_i – фиксированные позиции в блоках сообщения и ключе. Чем больше p отклоняется от 0.5, тем более подходящим считается уравнение.

Это уравнение означает, что если выполнить операцию XOR над некоторыми битами незашифрованного сообщения и над некоторыми битами зашифрованного сообщения, получится бит, представляющий собой XOR некоторых битов ключа. Это называется линейным приближением, которое может быть верным с вероятностью p.

Уравнения составляются следующим образом. Вычисляются значения левой части для большого числа пар соответствующих фрагментов незашифрованного и зашифрованного блоков. Если результат оказывается равен нулю более чем в половине случаев, то полагают, что $K[\gamma_1, \gamma_2, \ldots \gamma_c] = 0$. Если в большинстве случаев получается 1, полагают, что $K[\gamma_1, \gamma_2, \ldots \gamma_c] = 1$. Таким образом получают систему уравнений, решением которой является ключ.

Как и в случае дифференциального криптоанализа, результаты линейного криптоанализа должны учитываться при разработке алгоритмов симметричного криптоанализа.

Используемые критерии при разработке алгоритмов

Принимая во внимание перечисленные требования, обычно считается, что алгоритм симметричного шифрования должен:

- Манипулировать данными в больших блоках, предпочтительно размером 16 или 32 бита.
- Иметь размер блока 64 или 128 бит.
- Иметь масштабируемый ключ до 256 бит.
- Использовать простые операции, которые эффективны на микропроцессорах, т.е. исключающее или, сложение, табличные подстановки, умножение по модулю. Не должно использоваться сдвигов переменной длины, побитных перестановок или условных переходов.
- Должна быть возможность реализации алгоритма на 8-битном процессоре с минимальными требованиями к памяти.
- Использовать заранее вычисленные подключи. На системах с большим количеством памяти эти подключи могут быть заранее вычислены для ускорения работы. В случае невозможности заблаговременного вычисления подключей должно произойти только замедление выполнения. Всегда должна быть возможность шифрования данных без каких-либо предварительных вычислений.
- Состоять из переменного числа итераций. Для приложений с маленькой длиной ключа нецелесообразно применять большое число итераций для противостояния дифференциальным и другим атакам. Следовательно, должна быть возможность уменьшить число итераций без потери безопасности (не более чем уменьшенный размер ключа).
- По возможности не иметь слабых ключей. Если это невозможно, то количество слабых ключей должно быть минимальным, чтобы уменьшить вероятность случайного выбора одного из них. Тем не менее, все слабые ключи должны быть заранее известны, чтобы их можно было отбраковать в процессе создания ключа.

- Задействовать подключи, которые являются односторонним хэшем ключа. Это дает возможность использовать большие парольные фразы в качестве ключа без ущерба для безопасности.
- Не иметь линейных структур, которые уменьшают комплексность и не обеспечивают исчерпывающий поиск.
- Использовать простую для понимания разработку. Это дает возможность анализа и уменьшает закрытость алгоритма.

Большинство блочных алгоритмов основано на использовании сети Фейштеля, все имеют плоское пространство ключей, с возможным исключением нескольких слабых ключей.

Алгоритм DES

Принципы разработки

Самым распространенным и наиболее известным алгоритмом симметричного шифрования является DES (Data Encryption Standard). Алгоритм был разработан в 1977 году, в 1980 году был принят NIST (National Institute of Standards and Technolody США) в качестве стандарта (FIPS PUB 46).

DES является классической сетью Фейштеля с двумя ветвями. Данные шифруются 64-битными блоками, используя 56-битный ключ. Алгоритм преобразует за несколько раундов 64-битный вход в 64-битный выход. Длина ключа равна 56 битам. Процесс шифрования состоит из четырех этапов. На первом из них выполняется начальная перестановка (IP) 64-битного исходного текста (забеливание), во время которой биты переупорядочиваются в соответствии со стандартной таблицей. Следующий этап состоит из 16 раундов одной и той же функции, которая использует операции сдвига и подстановки. На третьем этапе левая и правая половины выхода последней (16-й) итерации меняются местами. Наконец, на четвертом этапе выполняется перестановка IP^{-1} результата, полученного на третьем этапе. Перестановка IP^{-1} инверсна начальной перестановке.

Рис. 2.4. Общая схема DES

Справа на рисунке показан способ, которым используется 56-битный ключ. Первоначально ключ подается на вход функции перестановки. Затем для каждого из 16 раундов подключ K_i является комбинацией левого циклического сдвига и перестановки. Функция перестановки одна и та же для каждого раунда, но подключи K_i для каждого раунда получаются разные вследствие повторяющегося сдвига битов ключа.

Шифрование
Начальная перестановка

Начальная перестановка и ее инверсия определяются стандартной таблицей. Если M – это произвольные 64 бита, то $X = IP(M)$ – переставленные 64 бита. Если применить обратную функцию перестановки $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, то получится первоначальная последовательность битов.

Последовательность преобразований отдельного раунда

Теперь рассмотрим последовательность преобразований, используемую в каждом раунде.

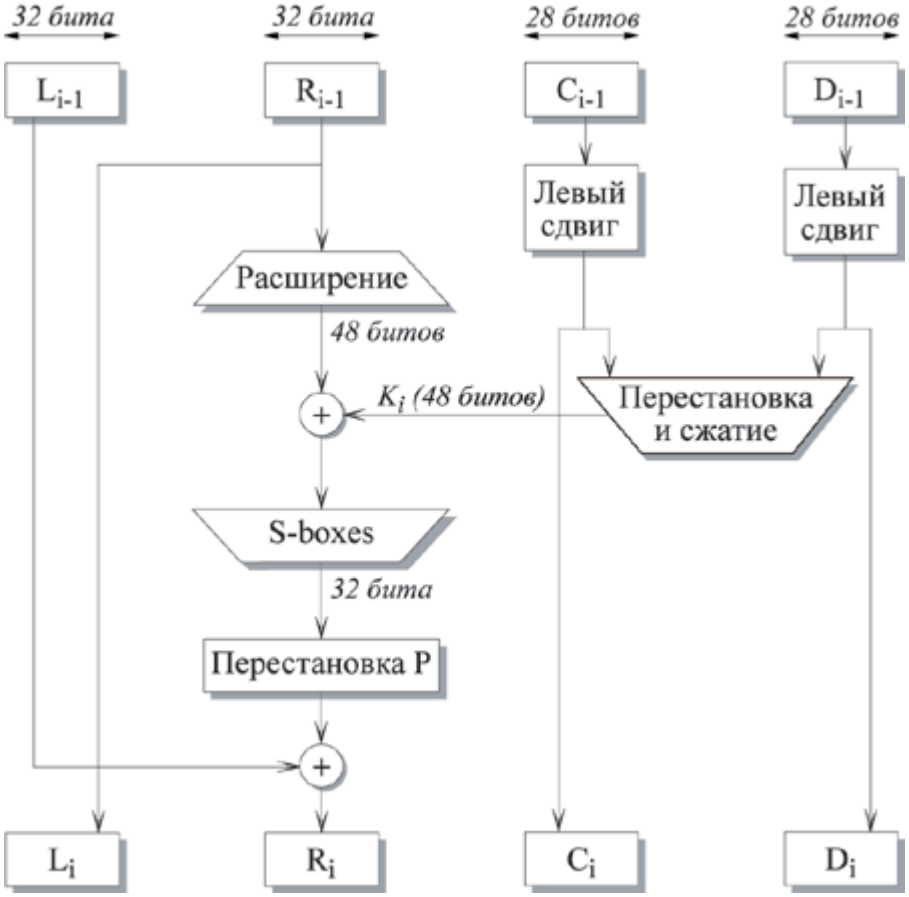

Рис. 2.5. I-ый раунд DES

64-битный входной блок проходит через 16 раундов, при этом на каждой итерации получается промежуточное 64-битное значение. Левая и правая части каждого промежуточного значения трактуются как отдельные 32-битные значения, обозначенные L и R. Каждую итерацию можно описать следующим образом:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

Где $\oplus$ обозначает операцию XOR.

Таким образом, выход левой половины L_i равен входу правой половины R_{i-1}. Выход правой половины R_i является результатом применения операции XOR к L_{i-1} и функции F, зависящей от R_{i-1} и K_i.

Рассмотрим функцию F более подробно.

R_i, которое подается на вход функции F, имеет длину 32 бита. Вначале R_i расширяется до 48 битов, используя таблицу, которая определяет перестановку плюс расширение на 16 битов. Расширение происходит следующим образом. 32 бита разбиваются на группы по 4 бита и затем расширяются до 6 битов, присоединяя крайние биты из двух соседних групп. Например, если часть входного сообщения

. . . efgh ijkl mnop . . .

то в результате расширения получается сообщение

. . . defghi hijklm lmnopq . . .

После этого для полученного 48-битного значения выполняется операция XOR с 48-битным подключом K_i. Затем полученное 48-битное значение подается на вход функции подстановки, результатом которой является 32-битное значение.

Подстановка состоит из восьми S-boxes , каждый из которых на входе получает 6 бит, а на выходе создает 4 бита. Эти преобразования определяются специальными таблицами. Первый и последний биты входного значения S-box определяют номер строки в таблице, средние 4 бита определяют номер столбца. Пересечение строки и столбца определяет 4-битный выход. Например, если входом является 011011, то номер строки равен 01 (строка 1) и номер столбца равен 1101 (столбец 13). Значение в строке 1 и столбце 13 равно 5, т.е. выходом является 0101.

Далее полученное 32-битное значение обрабатывается с помощью перестановки P , целью которой является максимальное переупорядочивание битов, чтобы в следующем раунде шифрования с большой вероятностью каждый бит обрабатывался другим S-box.

Создание подключей

Ключ для отдельного раунда K_i состоит из 48 битов. Ключи K_i получаются по следующему алгоритму. Для 56-битного ключа, используемого на входе алгоритма, вначале выполняется перестановка в соответствии с таблицей Permuted Choice 1 (PC-1). Полученный 56-битный ключ разделяется на две 28-битные части, обозначаемые как C_0 и D_0 соответственно. На каждом раунде C_i и D_i независимо циклически сдвигаются влево на 1 или 2 бита, в зависимости от номера раунда. Полученные значения являются входом следующего раунда. Они также представляют собой вход в Permuted Choice 2 (PC-2), который создает 48-битное выходное значение, являющееся входом функции $F(R_{i-1}, K_i)$.

Дешифрование

Процесс дешифрования аналогичен процессу шифрования. На входе алгоритма используется зашифрованный текст, но ключи K_i используются в обратной последовательности. K_{16} используется на первом раунде, K_1 используется на последнем раунде. Пусть выходом *i*-ого раунда шифрования будет $L_i \| R_i$. Тогда соответствующий вход (16-*i*)-ого раунда дешифрования будет $R_i \| L_i$.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки IP^{-1} был $R_{16} \| L_{16}$. Выходом этой стадии является незашифрованный текст.

Проверим корректность процесса дешифрования. Возьмем зашифрованный текст и ключ и используем их в качестве входа в алгоритм. На первом шаге выполним начальную перестановку IP и получим 64-битное значение $L_0^d \| R_0^d$. Известно, что IP и IP^{-1} взаимнообратны. Следовательно

$$L_0^d \| R_0^d = IP(\textit{зашифрованный текст})$$

$$\textit{Зашифрованный текст} = IP^{-1}(R_{16} \| L_{16})$$

$$L_0^d \| R_0^d = IP(IP^{-1}(R_{16} \| L_{16})) = R_{16} \| L_{16}$$

Таким образом, вход первого раунда процесса дешифрования эквивалентен 32-битному выходу 16-ого раунда процесса шифрования, у которого левая и правая части записаны в обратном порядке.

Теперь мы должны показать, что выход первого раунда процесса дешифрования эквивалентен 32-битному входу 16-ого раунда процесса шифрования. Во-первых, рассмотрим процесс шифрования. Мы видим, что

$$L_{16} = R_{15}$$

$$R_{16} = L_{15} \oplus F(R_{15}, K_{16})$$

При дешифровании:

$$L_1^d = R_0^d = L_{16} = R_{15}$$

$$R_1^d = L_0^d \oplus F(R_0^d, K_{16}) = R_{16} \oplus F(R_0^d, K_{16}) = (L_{15} \oplus F(R_{15}, K_{16})) \oplus F(R_{15}, K_{16})$$

XOR имеет следующие свойства:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

$$D \oplus D = 0$$

$$E \oplus 0 = E$$

Таким образом, мы имеем $L_1^d = R_{15}$ и $R_1^d = L_{15}$. Следовательно, выход первого раунда процесса дешифрования есть $L_{15}\|R_{15}$, который является перестановкой входа 16-го раунда шифрования. Легко показать, что данное соответствие выполняется все 16 раундов. Мы можем описать этот процесс в общих терминах. Для *i*-ого раунда шифрующего алгоритма:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

Эти равенства можно записать по-другому:

$$R_{i-1} = L_i$$

$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i) = R_i \oplus F(L_i, K_i)$$

Таким образом, мы описали входы *i*-ого раунда как функцию выходов.

Выход последней стадии процесса дешифрования есть $R_0\|L_0$. Чтобы входом IP^{-1} стадии было $R_0\|L_0$, необходимо поменять местами левую и правую части. Но

$$IP^{-1}(R_0\|L_0) = IP^{-1}(IP(\textit{незашифрованный текст})) = \textit{незашифрованный текст}$$

Т.е. получаем незашифрованный текст, что и демонстрирует возможность дешифрования DES.

Проблемы DES

Так как длина ключа равна 56 битам, существует 2^{56} возможных ключей. На сегодня такая длина ключа недостаточна, поскольку допускает успешное применение лобовых атак. Альтернативой DES можно считать тройной DES, IDEA, а также алгоритм Rijndael, принятый в качестве нового стандарта на алгоритмы симметричного шифрования.

Также без ответа пока остается вопрос, возможен ли криптоанализ с использованием существующих характеристик алгоритма DES. Основой алгоритма являются восемь таблиц подстановки, или S-boxes, которые применяются в каждой итерации. Существует опасность, что эти S-boxes конструировались таким образом, что криптоанализ возможен для взломщика, который знает слабые места S-boxes. В течение многих лет обсуждалось как стандартное, так и неожиданное поведение S-boxes, но все-таки никому не удалось обнаружить их фатально слабые места.

Алгоритм тройной DES

В настоящее время основным недостатком DES считается маленькая длина ключа, поэтому уже давно начали разрабатываться различные альтернативы этому алгоритму шифрования. Один из подходов состоит в том, чтобы разработать новый алгоритм, и успешный тому пример – IDEA. Другой подход предполагает повторное применение шифрования с помощью DES с использованием нескольких ключей.

Недостатки двойного DES

Простейший способ увеличить длину ключа состоит в повторном применении DES с двумя разными ключами. Используя незашифрованное сообщение P и два ключа K_1 и K_2, зашифрованное сообщение C можно получить следующим образом:

$$C = E_{K2}[E_{K1}[P]]$$

Для дешифрования требуется, чтобы два ключа применялись в обратном порядке:

$$P = D_{K1}[D_{K2}[C]]$$

В этом случае длина ключа равна 56·2 = 112 бит.

Атака "встреча посередине"

Для приведенного выше алгоритма двойного DES существует так называемая атака "встреча посередине". Она основана на следующем свойстве алгоритма. Мы имеем

$$C = E_{K2}[E_{K1}[P]]$$

Тогда

$$X = E_{K1}[P] = D_{K2}[C]$$

Атака состоит в следующем. Требуется, чтобы атакующий знал хотя бы одну пару незашифрованный текст и соответствующий ему зашифрованный текст: (P,C). В этом случае, во-первых, шифруется P для всех возможных 2^{56} значений K_1. Этот результат запоминается в таблице, и затем таблица упорядочивается по значению X. Следующий шаг состоит в дешифровании C, с применением всех возможных 2^{56} значений K_2. Для каждого выполненного дешифрования ищется равное ему значение в первой таблице. Если соответствующее значение найдено, то считается, что эти

ключи могут быть правильными, и они проверяются для следующей известной пары незашифрованный текст, зашифрованный текст.

Если известна только одна пара значений незашифрованный текст, зашифрованный текст, то может быть получено достаточно большое число неверных значений ключей. Но если противник имеет возможность перехватить хотя бы две пары значений (незашифрованный текст – зашифрованный текст), то сложность взлома двойного DES фактически становится равной сложности взлома обычного DES, т.е. 2^{56}.

Тройной DES с двумя ключами

Очевидное противодействие атаке "встреча посередине" состоит в использовании третьей стадии шифрования с тремя различными ключами. Это поднимает стоимость лобовой атаки до 2^{168}, которая на сегодняшний день считается выше практических возможностей. Но при этом длина ключа равна 56·3 = 168 бит, что иногда бывает громоздко.

В качестве альтернативы предлагается метод тройного шифрования, использующий только два ключа. В этом случае выполняется последовательность зашифрование-расшифрование-зашифрование (EDE).

$$C = E_{K1}\left[D_{K2}\left[E_{K1}[P]\right]\right]$$

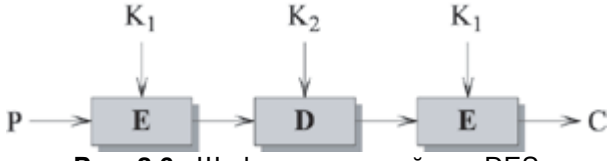

Рис. 2.6. Шифрование тройным DES

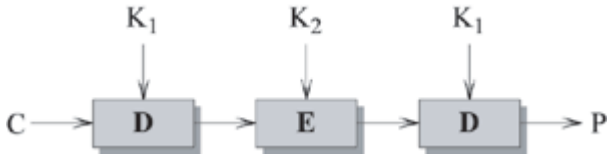

Рис. 2.7. Дешифрование тройным DES

Не имеет большого значения, что используется на второй стадии: шифрование или дешифрование. В случае использования дешифрования существует только то преимущество, что можно тройной DES свести к обычному одиночному DES, используя $K_1 = K_2$:

$$C = E_{K1}\left[D_{K1}\left[E_{K1}[P]\right]\right] = E_{K1}[P]$$

Тройной DES является достаточно популярной альтернативой DES и используется при управлении ключами в стандартах ANSI X9.17 и ISO 8732 и в PEM (Privacy Enhanced Mail).

Известных криптографических атак на тройной DES не существует. Цена подбора ключа в тройном DES равна 2^{112}.

Разработка Advanced Encryption Standard (AES)

Обзор процесса разработки AES

Инициатива в разработке AES принадлежит NIST. Основная цель состояла в создании федерального стандарта (FIPS), который бы описывал алгоритм шифрования, используемый для защиты информации как в государственном, так и в частном секторе.

Конкуренция среди финалистов была достаточно серьезной, и в результате длительного процесса оценки NIST выбрал Rijndael в качестве алгоритма AES. Мы кратко рассмотрим этот процесс и суммируем различные характеристики алгоритмов, которые были описаны на данном этапе. Следующий раздел представляет собой обзор разработки AES и обсуждение деталей алгоритмов.

Историческая справка

2 января 1997 года NIST объявил о начале разработки AES, и 12 сентября 1997 года были представлены официальные требования к алгоритмам. В этих требованиях указывалось, что целью NIST является разработка неклассифицированного, хорошо проанализированного алгоритма шифрования, доступного для широкого применения. Как минимум алгоритм должен быть симметричным и блочным и поддерживать длину блока 128 бит и длину ключа 128, 192 и 256 бит.

20 августа 1998 года NIST анонсировал пятнадцать кандидатов на алгоритм AES на первой конференции кандидатов AES (AES1), и было предложено прокомментировать их характеристики. Данные алгоритмы были разработаны промышленными и академическими кругами двенадцати стран. Вторая конференция кандидатов AES (AES2) была проведена в марте 1999 года с целью обсуждения результатов анализа предложенных алгоритмов. В августе 1999 года были представлены выбранные NIST пять финалистов. Ими стали MARS, RC6™, Rijndael, Serpent и Twofish.

Обзор финалистов

Все пять финалистов являются итерационными блочными алгоритмами шифрования: они определяют преобразование, которое повторяется определенное число раз над блоком шифруемых или дешифруемых данных. Шифруемый блок данных называется plaintext; зашифрованный plaintext называется ciphertext. Для дешифрования в качестве обрабатываемого блока данных используется ciphertext. Каждый финалист также определяет метод создания серии ключей из исходного ключа, называемый управлением ключом. Полученные ключи называются подключами. Функции раунда используют в качестве входа различные подключи для конкретного блока данных.

У каждого финалиста первая и последняя криптографические операции являются некоторой формой перемешивания подключей и блока данных. Такие операции, используемые на начальном шаге первого раунда и заключительном шаге последнего раунда, называются pre- и post-забеливанием (whitening) и могут быть определены отдельно.

Существуют также некоторые другие технические особенности финалистов. Четыре финалиста определяют таблицы подстановки, называемые S-box: AxB-битный S-box заменяет A входных битов на B выходных битов. Три финалиста определяют функции раунда, являющиеся сетью Фейштеля. В классической сети Фейштеля одна половина блока данных используется для модификации другой половины блока данных, затем половины меняются местами. Два финалиста не используют сеть Фейштеля, в каждом раунде обрабатывают параллельно весь блок данных, применяя подстановки и линейные преобразования; таким образом, эти два финалиста являются примерами алгоритмов, использующих линейно-подстановочное преобразование.

Далее рассмотрим каждый из алгоритмов в алфавитном порядке; профили и оценки будут представлены в следующих разделах.

MARS выполняет последовательность преобразований в следующем порядке: сложение с ключом в качестве pre-whitening, 8 раундов прямого перемешивания без использования ключа, 8 раундов прямого преобразования с использованием ключа, 8 раундов обратного преобразования с использованием ключа, 8 раундов обратного перемешивания без использования ключа и вычитание ключа в качестве post-whitening. 16 раундов с использованием ключа называются криптографическим ядром. Раунды без ключа используют два 8x16- битных S-boxes и операции сложения и XOR. В дополнение к этим элементам раунды с ключом используют 32-битное умножение ключа, зависимые от данных циклические сдвиги и добавление ключа. Как раунды перемешивания, так и раунды ядра являются раундами модифицированной сети Фейштеля, в которых четверть блока данных используется для изменения остальных трех четвертей блока данных. MARS предложен корпорацией IBM.

RC6 является параметризуемым семейством алгоритмов шифрования, основанных на сети Фейштеля; для AES было предложено использовать 20 раундов. Функция раунда в RC6 задействует переменные циклические сдвиги, которые определяются квадратичной функцией от данных. Каждый раунд также включает умножение по модулю 32, сложение, XOR и сложение с ключом. Сложение с ключом также используется для pre- и pos-whitening. RC6 был предложен лабораторией RSA.

Rijndael представляет собой алгоритм, использующий линейно-подстановочные преобразования и состоящий из 10, 12 или 14 раундов в зависимости от длины ключа. Блок данных, обрабатываемый с использованием Rijndael, делится на массивы байтов, и каждая операция шифрования является байт-ориентированной. Функция раунда Rijndael состоит из четырех слоев. В первом слое для каждого байта применяется S-box размерностью 8x8 бит. Второй и третий слои являются линейными перемешиваниями, в которых строки рассматриваются в качестве сдвигаемых массивов и столбцы перемешиваются. В четвертом слое выполняется операция XOR байтов подключа и каждого байта массива. В последнем раунде перемешивание столбцов опущено. Rijndael предложен Joan Daemen (Proton World International) и Vincent Rijmen (Katholieke Universiteit Leuven).

Serpent является алгоритмом, использующим линейно-подстановочные преобразования и состоящим из 32 раундов. Serpent также определяет некриптографические начальную и заключительную перестановки, которые облегчают альтернативный режим реализации, называемый bitslice. Функция раунда состоит из трех слоев: операция XOR с ключом, 32-х параллельное применение одного из восьми фиксированных S-boxes и линейное преобразование. В последнем раунде слой XOR с ключом заменен на линейное преобразование. Serpent предложен Ross Anderson (University of Cambridge), Eli Biham (Technion) и LarsKnudsen (University of California San Diego).

Twofish является сетью Фейштеля с 16 раундами. Сеть Фейштеля незначительно модифицирована с использованием однобитных ротаций. Функция раунда влияет на 32-битные слова, используя четыре зависящих от ключа S-boxes, за которыми следуют фиксированные максимально удаленные отдельные матрицы в $GF(2^8)$, преобразование псевдо-Адамара и добавление ключа. Twofish был предложен Bruce Schneier, John Kelsey и Niels Ferguson (Counterpane Internet Security, Inc.), Doug Whiting (Hi/fn, Inc.), David Wagner (University of California Berkley) и Chris Hall (Princeton University).

При объявлении финалистов представители NIST предложили обсудить и прокомментировать алгоритмы. На третьей конференции кандидатов AES (AES3), состоявшейся в апреле 2000 года, представленные комментирии были рассмотрены. Период открытого обсуждения был завершен 15 мая 2000 года.

Принципы разработки и происхождение алгоритмов

Исторические корни, лежащие в основе принципов разработки, влияют на доверие к алгоритму и на анализ его безопасности. Это также применимо к составным элементам алгоритма, таким как S-boxes. Требуется много времени для разработки атак на неизвестные технологии, традиционные технологии обычно анализируются больше, особенно если уже существует атака на них. Например, сеть Фейштеля хорошо изучена, т.к. она применяется в DES, и три финалиста используют варианты этой конструкции. Другим элементом, способным повлиять на доверие к алгоритму, является разработка S-boxes, которые могут содержать различные скрытые "люки". Это обсуждается далее для каждого финалиста.

MARS: разнородная структура раунда MARS не исследована. Как раунд перемешивания, так и раунд ядра основаны на сети Фейштеля со значительными изменениями. MARS использует много различных операций, большинство из которых традиционны. Материал ключа и данные применяются для регулирования различных операций ротации. S-box создается детерминировано с учетом требуемых свойств; таким образом, спецификация MARS утверждает, хотя это маловероятно, что MARS содержит какую-либо структуру, которая может использоваться в качестве люка для атаки. Спецификация MARS не рассматривает какой-либо алгоритм в качестве своего предшественника.

RC6: разработка RC6 развивалась из разработки RC5, который анализировался несколько лет. Безопасность обоих алгоритмов основана на переменных ротациях как основной источник нелинейности; S-boxes в алгоритме не существует. Операция переменной ротации в RC6, в отличие от RC5, регулируется квадратичной функцией от данных. Управление ключом в RC6 и RC5 не отличается. Структура раунда RC6 является вариантом сети Фейштеля. Спецификация RC6 утверждает, что в RC6 не существует люков, потому что при вычислении ключа используется только часть RC6, которая математически хорошо изучена.

Rijndael: Rijndael является байт-ориентированным алгоритмом шифрования, основанным на "Квадрате (Square)". Представляется, что Square-атака является отправной точкой для дальнейшего анализа. Типы операций подстановки и перестановки, используемые в Rijndael, являются стандартными. S-box имеет математическую структуру, основанную на комбинации инверсии в поле Галуа и афинного преобразования. Хотя данная математическая структура может помочь в осуществлении атаки, структура не является скрытой. Спецификация Rijndael утверждает, что если есть подозрение, что S-box содержит люк, то S-box может быть заменен.

Serpent: Serpent является байт-ориентированным алгоритмом. Типы операций подстановки и перестановки являются стандартными. S-boxes создаются детерминировано, как и в случае DES, и обладают теми же свойствами; спецификация Serpent устанавливает, что при такой конструкции нет опасения, что существуют люки. Спецификация Serpent не рассматривает какой-либо алгоритм в качестве своего предшественника.

Twofish: Twofish использует незначительную модификацию сети Фейштеля. Спецификация не рассматривает какой-либо алгоритм в качестве своего предшественника, но существует несколько предшествующих алгоритмов, которые разделяют важные свойства Twofish, такие как зависящие от ключа S-boxes и подходы к их разработке. Спецификация Twofish предполагает, что Twofish не имеет люков и доказывает это утверждение несколькими аргументами, включая переменные S-boxes.

Краткая характеристика финалистов

Как уже отмечалось, ни против одного из финалистов не существует общих атак. Однако определение уровня безопасности, обеспечиваемого финалистами, является достаточно приблизительным. Суммируем некоторые известные характеристики безопасности финалистов.

MARS показывает высокую степень резерва безопасности. Кратко охарактеризовать MARS трудно, потому что фактически MARS реализует два различных типа раунда. MARS даже критиковали за сложность, которая может препятствовать анализу его безопасности.

RC6 показал адекватный резерв безопасности. Однако RC6 критиковали за небольшой резерв безопасности по сравнению с другими финалистами. С другой стороны, все высоко оценили простоту RC6, облегчающую анализ безопасности. RC6 произошел от RC5, который уже достаточно хорошо проанализирован.

Rijndael показал адекватный резерв безопасности. Резерв безопасности довольно трудно измерить, потому что число раундов изменяется в зависимости от длины ключа. Rijndael критиковался по двум направлениям: что его резерв безопасности меньше, чем у других финалистов, и что его математическая структура может привести к атакам. Тем не менее, его структура достаточно проста и обеспечивает возможность анализа безопасности.

Serpent показал значительный резерв безопасности. Serpent также имеет простую структуру, безопасность которой легко проанализировать.

Twofish показал высокий резерв безопасности. Поскольку Twofish использует зависящую от ключа функцию раунда, для него замечания о резерве безопасности могут иметь меньшее значение, чем для других финалистов. Зависимость S-boxes Twofish только от k/2 битов энтропии в случае k-битного ключа позволяет сделать вывод, что Twofish может быть подвегнут divide-and-conquer-атаке, хотя такая атака не найдена. Twofish был подвергнут критике за сложность, которая затрудняет его анализ.