Данный материал является фрагментом электронного учебника
по информационной безопасности и может обновляться.
При цитировании рекомендуется использовать ссылку:

Амелин Р. В. Информационная безопасность. Конспект лекций // Амелин Р.В. Лаборатория преподавателя [Электронный ресурс]. URL: rv-lab.ru (2017).

Лекция 2. Современные симметричные шифры

2.1. Блочные шифры

Блочный шифр предназначен для шифрования небольших блоков фиксированной длины. Длина блока большинства современных шифров варьируется от 64 до 512 бит. На выходе получается блок зашифрованного текста той же длины. Эта длина называется *размером блока*. Если необходимо зашифровать таким шифром сообщение произвольной длины, его разбивают на блоки, после чего каждый блок шифруется отдельно с использованием одного из режимов, о которых будет рассказано далее.

Современные блочные шифры имеют одинаковую структуру, смысл которой заключается в том, что сообщение проходит несколько раундов шифрования, т.е. сначала оно шифруется, затем результат шифруется снова и так далее – всего n раз. Каждый раз используется новый ключ, называемый ключом раунда – все ключи раунда генерируются по определенному алгоритму из общего секретного ключа. Структура блочного шифра изображена на рис. 0.1.

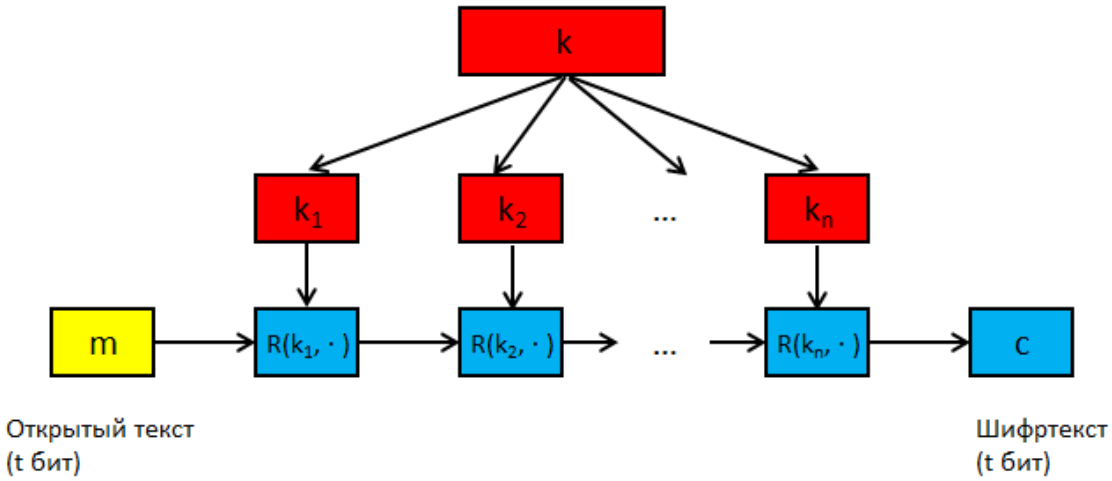

Рис. 0.1. Структура блочного шифра

Таким образом, различные блочные шифры отличаются следующими параметрами:

- Числом раундов n (с каждым новым раундом обработки надежность шифра повышается, но уменьшается скорость операций шифрования/дешифрования);
- Размером блока t (чем выше размер блока, тем больше надежность шифра, но скорость при этом снижается);
- Размером ключа k (чем он больше, тем более устойчив алгоритм к полному перебору);
- Алгоритмом генерации ключей раунда $k_1 \ldots k_n$.
- Функцией раунда R. Именно эта функция является «сердцем» блочного шифра, определяющим параметры его надежности. Практически все функции раунда современных блочных алгоритмов шифрования строятся по одной из двух моделей: *схема Файстеля* или *сеть подстановок и перестановок*.

При проведении криптоанализа шифра чаще всего пытаются изобрести успешную атаку для его версии с небольшим количеством раундов, а затем усовершенствовать ее до атаки на весь шифр.

2.2. Схема Файстеля

Большинство современных алгоритмов имеют структуру, аналогичную структуре шифра Файстеля, разработанного в 1973 году.

Шифр Файстеля создавался как пример практической реализации идеи Клода Шеннона: *надежный алгоритм шифрования должен удовлетворять двум свойствам: диффузии и коффузии.*

Диффузия – каждый бит открытого текста должен влиять на каждый бит зашифрованного текста. Суть диффузии заключается в рассеянии статистических характеристик открытого текста внутри шифрованного текста.

Конфузия – отсутствие статистической взаимосвязи между ключом и зашифрованным текстом. Даже если противник определит какие-то статистические особенности зашифрованного текста, их должно оказаться недостаточно, чтобы получить любую информацию о ключе.

На вход алгоритма шифрования подается блок открытого текста, имеющий четную длину *2l* и ключ *K*. Блок разделяется на две равные части – правую R0 и левую L0. Далее эти части проходят m раундов обработки, после чего снова объединяются в зашифрованный текст.

Каждый *i*-й раунд состоит в генерации подключа K_i (на основе общего ключа K) и применении к блоку R_i некоторого зависящего от ключа преобразования *F*. Результат складывается с блоком L_i с помощью операции XOR (исключающее или) и получается блок R_{i+1}. Блок R_i без изменений берется в качестве блока L_{i+1}.

Функция F может быть необратимой (а может быть, $F^{-1}(x)$ невозможно вычислить алгоритмически). Независимо от выбраной функции процесс дешифрования принципиально ничем не отличается от процесса шифрования. Просто на вход подается зашифрованный текст, а ключи K_i вычисляются в обратном порядке. Это крайне удобно с точки зрения реализации (особенно аппаратной).

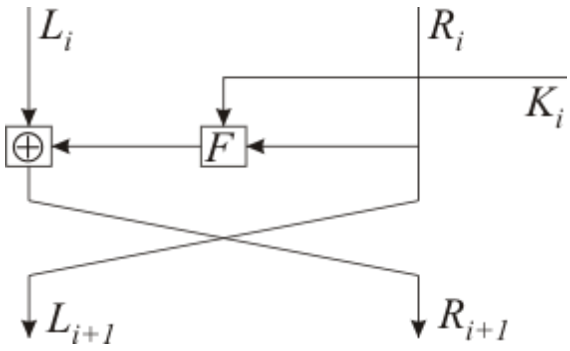

Рис. 2.2. Схема i-го раунда шифрования шифра Файстеля

Доказана теорема, что если функция F является безопасной, то трехраундовая схема Файстеля с функцией раунда F также безопасна.

Под безопасной функцией в криптологии понимается такая функция, которую нельзя по выходу отличить от истинно случайной функции. Представим себе множество всех функций, которые переводят n-битный блок в другой n-битный блок. Можно представить каждую функцию таблицей вход-выход (т.е. набором значений). Всего таких функций невообразимо гигантское число (для 128 битного блока это $2^{128^{2^{128}}}$). Среди них есть и те, которые заданы аналитически, то есть, некоторой формулой, правилом, например, функция *f(x) = x*. Зная это правило, мы легко вычисляем значение функции. Представим себе противника, который пытается отгадать это правило, подавая на вход функции различные значения и наблюдая за результатом. Так вот, если противник, совершая сколь угодно большое число попыток, не может это правило отгадать и научиться предсказывать значения нашей функции для тех входных данных, которые он еще не тестировал (другими словами не может отличить нашу функцию от истинно случайной функции, взятой из этого гигантского множества), то назовем такую функцию *безопасной*.

Шифр – это тоже функция, которая переводит открытый текст в зашифрованный, но к шифру предъявляется дополнительное требование: он должен быть обратимым. Это значит, что два разных открытых текста всегда переводятся в два разных зашифрованных текста (иначе мы не могли бы однозначно дешифровать сообщение). Следовательно, с учетом того, что размеры открытого и зашифрованного текста совпадают, каждому из возможных зашифрованных текстов соответствует ровно один открытый текст. Такая функция называется *перестановкой*. Построить безопасную перестановку намного сложнее, чем безопасную функцию, и схема Файстеля как раз решает эту задачу.

Алгоритм DES

Долгое время самым популярным алгоритмом симметричного шифрования являлся *DES* (Data Encrypting Standart), принятый в 1977 году. Этот алгоритм базируется на структуре шифра Файстеля с размером блока 64 бита и 56-битным ключом.

Преобразование F заключается в следующем:

1. 32-битовый блок R_i расширяется до 48 битов с помощью специальной таблицы путем дублирования некоторых 16 битов.

2. Полученный результат складывается с 48-битным подключом K_i операцией XOR.

3. Результат сложения разбивается на 8 шестибитовых блоков и каждый из них преобразуется с помощью одной из шести S-матриц.

4. Получившийся в итоге 32-битный блок подвергается жестко заданной в алгоритме перестановке.

S-матрицы

S-матрицы, ставшие популярными вместе с алгоритмом DES, превратились в один из криптографических примитивов – регулярно используемых строительных блоков шифров и других криптографических алгоритмов.

Идея S-матрицы очень проста. По сути, она представляет собой таблично заданную функцию: для каждого входного значения «вручную» указывается выходное значение.

Основная цель заключается в том, чтобы избежать использования в шифре аналитически заданных функций, в первую очередь – функций, допускающих линейные зависимости.

Пусть $m_1, m_2, \ldots m_n$ – входные биты, а $c_1, c_2 \ldots c_n$ – выходные биты некоторого преобразования. Если для некоторых i найдутся такие $a_{i1} \ldots a_{in}$, что наблюдается равенство $c_i = a_{i1}m_1 \oplus a_{i2}m_2 \oplus \ldots \oplus a_{in}m_n$ для всех входных сообщений[1], то будем говорить, что существует линейная зависимость между входными битами и выходным битом ci. Для того, чтобы найти эту зависимость (т.е. $a_{i1} \ldots a_{in}$) достаточно решить систему уравнений с n неизвестными.

Если для некоторого ключа K алгоритм шифрования допускает линейную зависимость, то для нахождения этой зависимости достаточно получить n образцов пар открытый текст + шифртекст (криптоанализ с известным открытым текстом). После этого расшифровывать сообщения, зашифрованные тем же ключом можно будет, даже не зная самого ключа.

Даже если линейная зависимость выполняется только для некоторых входных битов, проявляется только на уровне функции раунда и лишь с определенной вероятностью, этого достаточно, чтобы противник мог *проверять* некоторые предположения относительно неизвестного ему открытого текста, что является признаком ненадежности алгоритма шифрования. Криптоанализ, основанный на попытках найти линейную зависимость между битами шифртекста и битами открытого текста называется *линейным криптоанализом*.

S-матрица – это способ задать вручную функцию, которую нельзя задать аналитически (в том числе с помощью линейных зависимостей). Например, одна из матриц алгоритма DES выглядит следующим образом:

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

В DES S-матрицы задают преобразование 6-битного блока в 4-битный. Входной блок делится на 2 части: номер столбца (первый и последний бит) и номер строки (четыре средних бита). Число на пересечении соответствующих строки и столбца (в двоичной записи) и есть результат преобразования. То же самое можно было записать в виде простой таблицы из 64 строк, указав соответствие для каждого входного блока.

S-матрицы считались самой сомнительной частью алгоритма DES, поскольку создатели алгоритма не раскрыли принципы их заполнения (почему выбраны именно эти мат-

[1] Поскольку *aij* может быть равно 0 или 1, это означает, что бит *ci* может быть получен сложением по модулю 2 определенных входных битов.

рицы и не содержит ли алгоритм «тайных ходов»). Однако за годы попыток взлома этого алгоритма была выявлена незначительная слабость лишь одной из этих матриц.

2.3. Сеть подстановок и перестановок

Сеть подстановок и перестановок – вторая популярная схема построения современных блочных шифров (рис. 2.3).

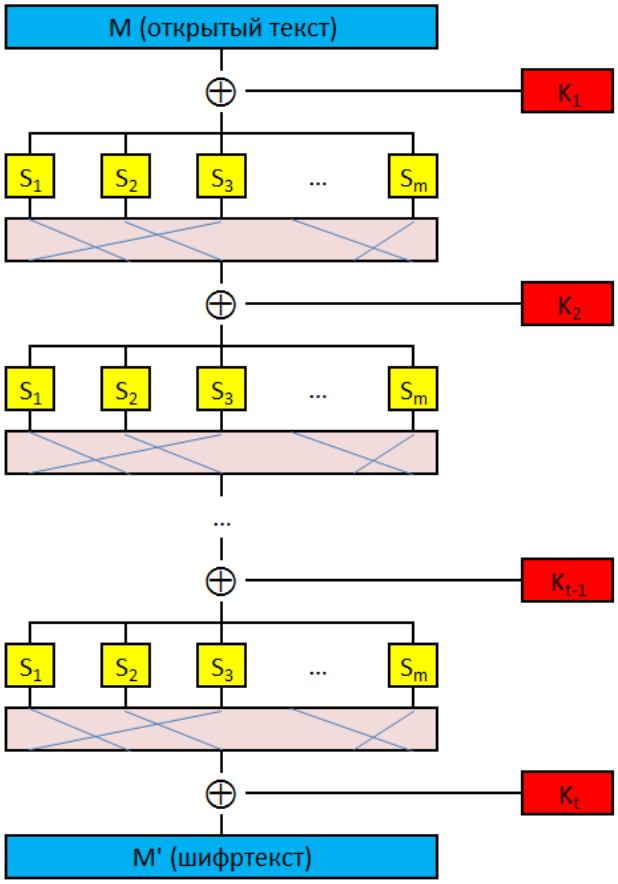

Рис. 2.3. Сеть подстановок и перестановок

На каждом раунде шифрования входной блок складывается по модулю 2 с ключом раунда, а затем разбивается на фрагменты (внутренние блоки), к каждому из которых применяется некоторая подстановка. После этого биты блока меняются местами – выполняется перестановка. Подстановки и перестановки являются фиксированными (жестко заданными для конкретного алгоритма) и не зависящими от ключа.

Подстановки (по аналогии с S-матрицами) как правило задаются в виде нелинейной табличной функции. Их называют S-подстановками.

Дешифрование осуществляется по той же схеме, но ключи раунда подаются в обратном порядке, а на каждом этапе используются обратные подстановки и перестановки. Соответственно, все подстановки и перестановки должны быть обратимыми.

При внешней простоте сеть подстановок и перестановок гарантированно обеспечивает лавинный эффект. Представим, что во входном блоке изменился всего один бит. В первом раунде шифрования это изменение затронет только одну подстановку, но на выхо-

де этой подстановки будут изменены уже несколько битов (в силу ее нелинейности). Измененные биты в результате воздействия перестановки разлетятся на несколько внутренних блоков второго раунда и попадут на вход уже нескольких подстановок. Через 3-4 раунда это изменение так или иначе затронет все внутренние блоки и скажется практически на каждом бите. Аналогичным образом обеспечивается свойство конфузии (изменение хотя бы одного бита ключа приведет к изменению хотя бы одного бита в ключах раунда, а после сложения с открытым текстом перед применением подстановок и перестановки мы получим тот же результат, что и при изменении бита открытого текста).

Алгоритм AES

Долгое время DES являлся федеральным стандартом шифрования США. Этот алгоритм показывает хороший *лавинный эффект* (изменение одного бита открытого текста или ключа приводит к изменению многих битов зашифрованного текста) и успешно противостоял многолетним попыткам взлома. Однако длина ключа в 56 битов при возросшей производительности ЭВМ сделала шифр потенциально уязвимым к перебору ключей, поэтому в 1997 году был объявлен конкурс на новый алгоритм, который должен был стать криптостандартом на ближайшие 10-20 лет.

Победитель конкурса был определен в 2000 году – им стал бельгийский шифр RIJNDAEL, который был переименован в AES (Advanced Encryption Standard).

AES, построен на основе сети подстановок и перестановок. В нем используется 128-битный блок и 8-битные S-подстановки (16 внутренних блоков). Длина ключа K составляет 128, 192, или 256 бит. Число раундов шифрования зависит от длины ключа (10, 12 и 14 раундов соответственно).

Среди других современных алгоритмов симметричного шифрования следует назвать шифры IDEA, Blowfish, RC5, CAST-128.

2.4. *Режимы использования блочных шифров*

Блочные шифры предназначены для преобразования блока небольшой фиксированной длины (64 бита в алгоритме DES, 128 бит в AES и т.д.). Чтобы зашифровать с помощью блочного шифра сообщение произвольной длины, могут использоваться несколько различных способов, которые называются ***режимами использования***.

*1. Режим электронной шифровальной книги (**ECB**[2]).*

Наиболее простой и естественный способ. Текст разбивается на блоки, и каждый блок шифруется с одним и тем же ключом (рис. 2.4). Проблема состоит в том, что два одинаковых блока открытого текста превращаются в одинаковые блоки шифртекста. Это значит, что противник, имея только зашифрованный текст, может получать некоторую информацию об открытом тексте, следовательно, шифр является небезопасным. Даже если противник не сможет расшифровать эти блоки, он будет знать, что они одинаковые.

[2] Electronic Codebook

Например, противник может отслеживать состояние зашифрованного файла на диске и наблюдать, какие его фрагменты остаются неизменными, а какие меняются. Если же размер блока небольшой (например, 64 бита), а открытый текст представляет собой, например, последовательность символов в формате UNCODE (32 бита на символ), то блочный шифр превращается, по сути, в шифр простой замены – и противник может воспользоваться знаниями о частоте встречаемости биграмм. Таким образом, режим электронной шифровальной книги использовать не рекомендуется.

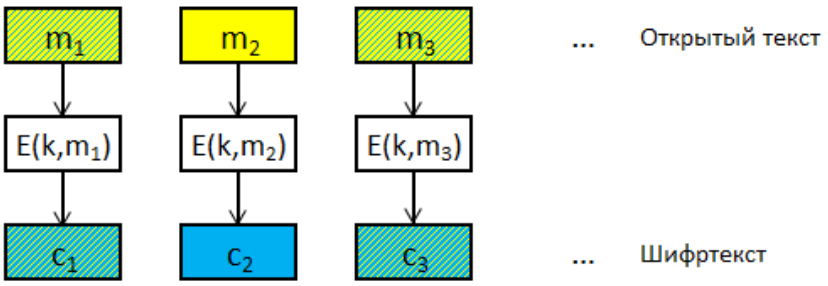

Рис. 2.4. Режим электронной кодовой книги

*2. Режим сцепления шифрованных блоков (**CBC**[3]).*

Каждый блок открытого текста перед шифрованием объединяется с помощью операции XOR с предыдущим блоком зашифрованного текста. Первый блок объединяется с некоторым заранее заданным *инициализационным вектором* (См. рис. 2.5). В результате одинаковые блоки открытого текста в зашифрованном виде будут различаться.

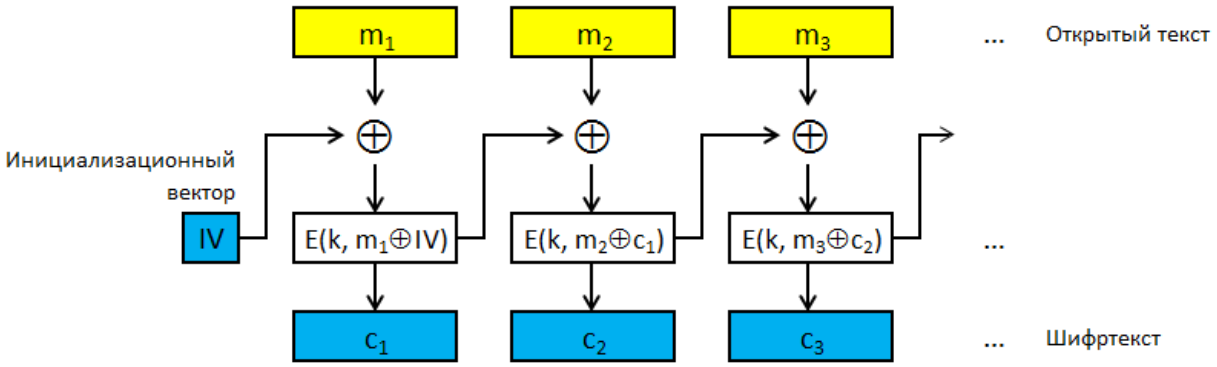

Рис. 2.5. Режим сцепления шифрованных блоков

В качестве инициализационного вектора может выбираться случайное значение (в этом случае его необходимо присоединить к шифртексту, чтобы сообщение можно было расшифровать), а можно генерировать по какому-то принципу, известному отправителю и получателю (например, использовать порядковый номер сообщения). Однако, если про-

[3] Cipher Block Chaining

тивник сможет *предсказывать* значение IV, данная схема становится уязвимой к атаке с избранным открытым текстом.

Оба рассмотренных режима требуют, чтобы открытый текст разбивался на блоки длины n. Однако длина открытого текста может быть не кратной n. В этом случае последний блок открытого текста дополняется[4] до нужной длины.

3. Режим счетчика (**CTR**[5]).

На вход алгоритма подается открытый текст, ключ и инициализационный вектор IV. Значение IV шифруется, результат складывается с первым блоком открытого текста по модулю 2. Затем значение IV увеличивается на единицу, и процедура повторяется (см. рис. 2.6). По сути, в режиме CTR на основе ключа и IV генерируется длинная последовательность бит, которая просто XOR-ится с открытым текстом, как это характерно для *потоковых шифров*.

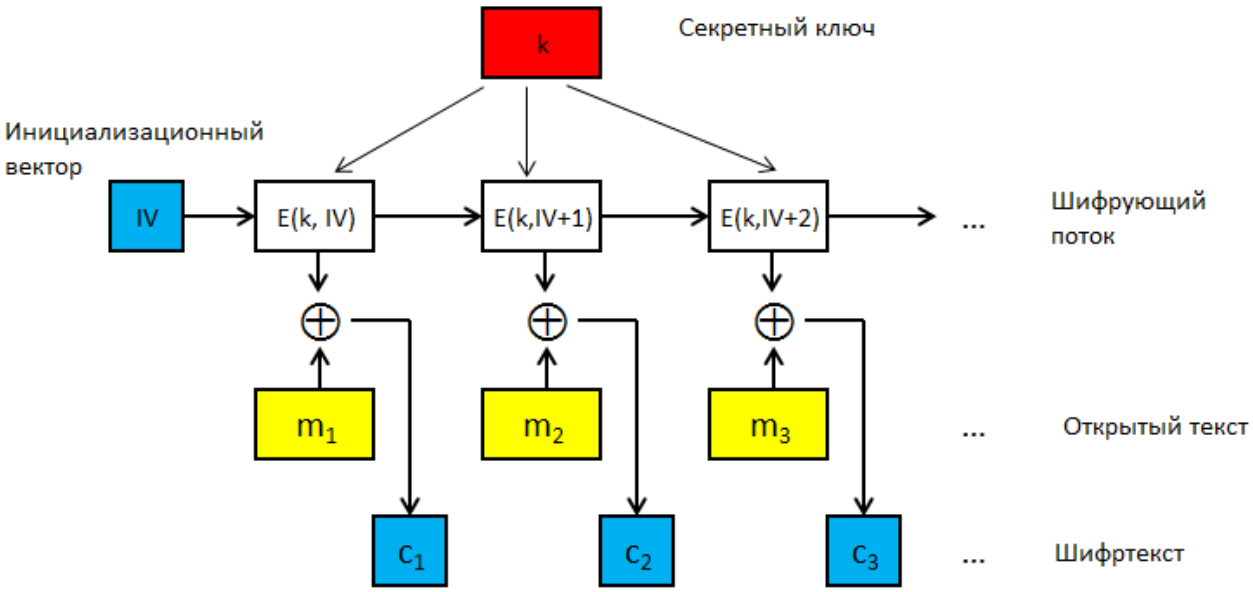

Рис. 2.6. Режим счетчика

Режим CTR имеет ряд плюсов. Во-первых, операция шифрования идентична операции дешифрования (а в основу вместо блочного шифра E может быть положена односторонняя функция F(k,m)). Во-вторых, эти операции могут быть распараллелены. Например, на четырехядерном процессоре могут одновременно выполняться операции шифрования сразу четырех блоков (в отличие от предыдущего режима, чтобы шифровать i-й блок текста, не надо дожидаться окончания шифрования i-1-го). В третьих, пропадает необходимость набивки. Необязательно, чтобы последний блок открытого текста имел длину n: мы можем взять для операции XOR столько битов шифрующей последовательности, сколько нам необходимо.

[4] Padding

[5] Counter mode

4.*Режим обратной связи по выходу (**OFB**[6]).*

Этот режим очень похож на предыдущий, но на каждом этапе генерации шифрующего потока входом функции шифрования служит не счетчик, а результат предыдущего этапа шифрования. Очевидно, что распараллелить вычисления в этом режиме не удастся.

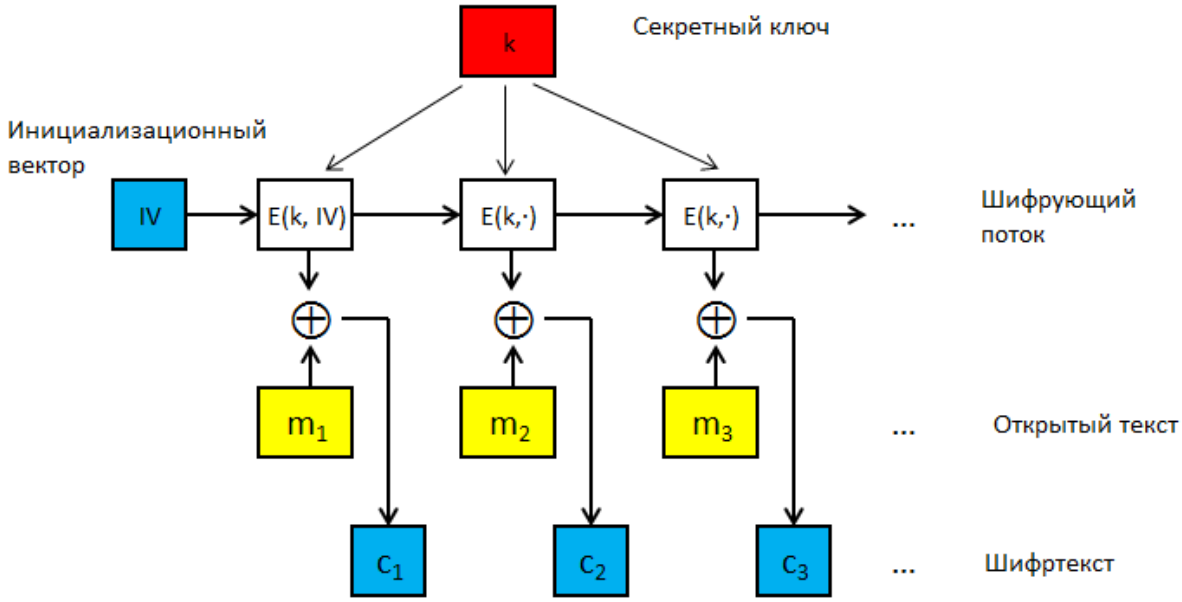

Рис. 2.7. Режим обратной связи по выходу

В последних двух режимах шифрующая последовательность, очевидно, будет одинаковой, если использовать одну и ту же пару ключ k и инициализационный вектор IV. Перехватив сообщения, зашифрованные такой последовательностью (обозначим ее t), противник может провести операцию сложения шифртекстов по модулю 2:

$c_1 \oplus c_2 = m_1 \oplus t \oplus m_2 \oplus t = m_1 \oplus m_2$

Таким образом, противник будет располагать суммой двух открытых текстов и если ему получится узнать (или сделать удачное предположение) фрагмент одного из них, то второй раскрывается тривиально. Поэтому основное правило использования режимов, основанных на применении инициализационного вектора, гласит: *пара <k, IV> должна быть уникальной*. Значит, каждое новое сообщение должно шифроваться с использованием нового IV, а зашифровав порядка 2^{32} сообщений, следует сменить ключ.

Заметим интересную особенность режима OFB. Если в шифрующей последовательности случайно встретятся два одинаковых блока (например, на i-м шаге шифрования мы получим в результате IV), то далее последовательность будет циклически повторяться. На самом деле, рано или поздно это, конечно, случится, ведь число блоков конечно. Но вероятность того, что это произойдет относительно быстро, обратно пропорциональна числу всевозможных блоков, то есть порядка $1/2^{128}$. Очень маленькая вероятность. Тем не менее, зашифровав достаточное количество блоков одного сообщения (порядка 16 Тб) следует сменить ключ или инициализационный вектор и начать следующее сообщение.

[6] Output feedback

2.5. Скремблирование

Одним из наиболее простых для понимания и реализации потоковых алгоритмов шифрования является скремблер. *Скремблерами* называются программные или аппаратные реализации алгоритма, позволяющего шифровать побитно непрерывные потоки информации. Сам скремблер представляет из себя набор бит, изменяющихся на каждом шаге по определенному алгоритму. После выполнения каждого очередного шага на его выходе появляется шифрующий бит – либо 0, либо 1, который накладывается на текущий бит информационного потока операцией XOR.

Рассмотрим пример простейшего скремблера. Он задается двумя битовыми последовательностями равной длины, одна из которых называется *ключом* (начальной последовательностью), а вторая собственно скремблером (часто вторая последовательность является фиксированной для конкретной аппаратной или программной реализации, а ключ выбирается как в обычном симметричном шифровании). Чем больше длина ключа (и скремблера), тем более надежным будет алгоритм.

Начальная последовательность (ключ) накладывается на скремблер, представляющий собой маску: выбираются только те биты последовательности, позициям которых соответствует единица в скремблере. Далее выбранные биты складываются между собой операцией XOR. Получается новый бит, который записывается в начало (слева) ключа. Последний (правый) бит ключа становится первым символом кодирующей последовательности и отбрасывается. Таким образом, происходит сдвиг ключа и генерация одного бита. Этот бит накладывается на первый бит исходного текста операцией XOR и получается первый бит зашифрованного текста. После этого цикл повторяется.

Пусть необходимо зашифровать сообщение 00111 скремблером 101 с ключом 011. Вычисляется сумма по модулю 2 первого и третьего бита ключа: 0 ⊕ 1 = 1. Этот бит становится новым первым битом ключа, а последний бит ключа (1) становится битом шифрующей последовательности. Вычисляем первый бит зашифрованного текста: 1 ⊕ 0 = 1. Далее повторяем процесс, но уже с ключом 101. Шифрование всего сообщения показано на рисунке 2.8.

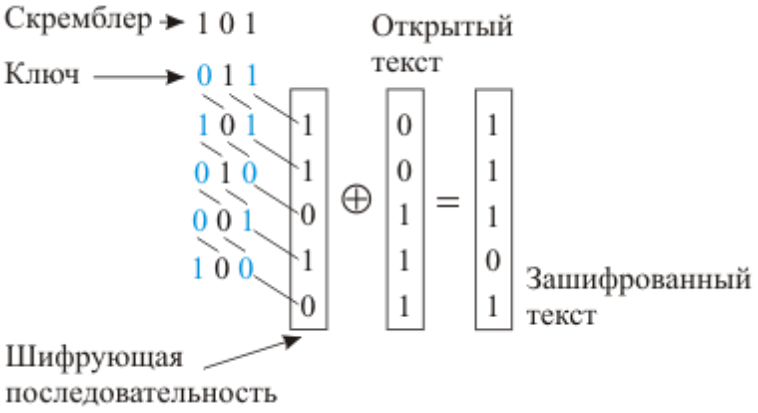

Рис. 2.8. Скремблирование последовательности 00111 скремблером 101 с ключом 011

2.6. Основные разновидности криптоанализа симметричных шифров

Классическая задача ***криптоанализа*** – получение открытого текста по зашифрованному тексту, не располагая при этом ключом. Часто под этим подразумевается нахождение ключа.

Некоторые методы криптоанализа шифров обсуждались выше при рассмотрении соответствующих алгоритмов шифрования.

Одна из популярных классификаций методов криптоанализа – по тем данным, которыми должен располагать аналитик. Соответственно выделяют:

- *Анализ только шифрованного текста*. Криптоаналитику известны алгоритм шифрования и зашифрованный текст.
- *Анализ с известным открытым текстом.* Криптоаналитик дополнительно располагает несколькими парами соответствующих друг другу фрагментов открытого и зашифрованного текста, созданного с одним и тем же ключом.
- *Анализ с избранным открытым текстом.* Криптоаналитик имеет возможность выбирать открытый текст для шифрования (т.е. располагает интересующим его открытым текстом и соответствующим ему зашифрованным).
- *Анализ с избранным зашифрованным текстом.* Криптоаналитик имеет возможность выбирать получать открытый текст для некоторых интересующих его образцов зашифрованного текста.
- *Анализ с избранным текстом.* Возможности криптоаналитика включают два предыдущих случая.

Существуют также отдельные методы криптоанализа, широко применяющиеся для взлома современных шифров. Перечислим наиболее известные из них.

Статистический криптоанализ. Основан на подсчете частоты встречаемости отдельных символов (или групп символов) в зашифрованном тексте. Подходит для взлома симметричных подстановочных алгоритмов.

Дифференциальный криптоанализ. Разновидность анализа с избранным открытым текстом. Используется при взломе современных симметричных алгоритмов шифрования, в которых текст последовательно проходит несколько раундов преобразований (в частности, известен алгоритм дифференциального криптоанализа шифра DES). Метод основан на прослеживании изменения схожести между двумя текстами. Выбирается пара незашифрованных текстов с определенным отличием (X), после чего 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. Эта вероятность и присваивается данному подключу раунда. Затем выбирается подключ с наибольшей вероятностью. Процесс повторяется для всех раундов. Цель – определить таким образом все подключи данного ключа (сам ключ при этом может остаться неизвестным).

Линейный криптоанализ. Применяется для взлома блочных симметричных шифров. В основе лежит понятие линейного приближения – предположение о том, что если выполнить операцию XOR над некоторыми битами открытого текста, затем над некоторыми битами шифртекста, а затем над результатами, получится бит, который представляет собой XOR некоторых бит ключа. Если это предположение верно с вероятностью выше ½, то на основе большого числа известных пар открытый текст/зашифрованный текст можно с удовлетворительной вероятностью определить значения отдельных битов ключа.

2.7. Проблемы симметричных алгоритмов

Все алгоритмы симметричного шифрования имеют общую проблему, проистекающую из того обстоятельства, что *отправитель и получатель сообщения должны обладать одним и тем же ключом*. При этом предполагается, что у них нет абсолютно надежного канала связи, поскольку в противном случае в шифровании бы не было нужды.

Если столетие назад эта проблема вполне решалась, например, путем личной встречи. Но в настоящее время, когда интенсивность обмена информацией возросла в сотни раз и автоматизированы практически все сферы человеческой деятельности, это невозможно. Необходимость в срочной конфиденциальной переписке может возникнуть у деловых партнеров, живущих в разных странах (и, может быть, даже не знакомых лично). Интернет-банкинг позволяет управлять своим банковским счетом, не выходя из дома, но при этом все операции должны быть конфиденциальными, следовательно, весь поток данных между банком и клиентом должен быть зашифрован. При этом ключи шифрования должны регулярно меняться, поскольку обмен даже несколькими сообщениями с одним ключом уменьшает надежность шифрования.

Таким образом, для симметричных алгоритмов характерна *проблема обмена ключами*. В настоящее время существует несколько способов ее решения, которые будут рассматриваться далее.

Кроме того, возникает проблема управления большим количеством ключей, поскольку отдельный ключ необходим для каждой пары «отправитель-получатель». Если группа из n человек желает обмениваться конфиденциальными сообщениями, понадобится $O(n^2)$ ключей (n-1 ключ каждому). Если же группа будет использовать один общий ключ, то его компрометация (утечка) у одного члена скомпрометирует переписку всей группы.

Эти проблемы и привели к появлению в середине XX века принципиально нового класса алгоритмов шифрования – алгоритмов шифрования с открытым ключом, которые будут рассмотрены далее.