Л.К. Бабенко, Е.А. Ищукова

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ: СИММЕТРИЧНОЕ ШИФРОВАНИЕ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ: СИММЕТРИЧНОЕ ШИФРОВАНИЕ

учебное пособие

Таганрог
Издательство Южного федерального университета
2015

УДК 004.056.55(075.8)

Печатается по решению редакционно-издательского совета
Южного федерального университета
(протокол №3 от 23 ноября 2015 года)

Рецензенты: Макаревич О. Б., доктор технических наук, профессор, научный руководитель Южно-Российского регионального центра по проблемам информационной безопасности в системе высшей школы ЮФУ; Ромм Я. Е., доктор технических наук, профессор, зав. кафедрой информатики Таганрогского института им. А.П. Чехова (филиал) «Ростовского государственного экономического университета» (РИНХ)

Бабенко Л.К., Ищукова Е.А. Криптографическая защита информации: симметричное шифрование: учебное пособие. – Таганрог: Изд-во ЮФУ, 2015. – 219 с.

Настоящее учебное пособие посвящено изучению основных аспектов современной криптографии, а именно ее большому разделу – симметричным блочным шифрам. Большое количество наглядных примеров позволит освоить основные принципы применения криптографических алгоритмов. А вопросы для самоконтроля и задачи для самостоятельного решения будут способствовать закреплению изученного материала. Учебное пособие по курсу "Криптографические методы защиты информации" рассчитано на студентов, магистрантов и аспирантов направлений 10.03.01 «Информационная безопасность», 10.05.02 «Информационная безопасность телекоммуникационных систем», 10.05.03 «Информационная безопасность автоматизированных систем». Изученный материал позволит самостоятельно применять полученные знания на практике в области криптографии и в области криптоанализа.

Табл. 39. Ил. 105. Библиогр. 27 назв

Содержание

Введение

Не секрет, что стремление защитить свои интересы было присуще человеку с давних пор. Еще в древности он использовал различные варианты кодирования информации, изобретал устройства, которые бы способствовали созданию более стойких шифров и при этом обеспечивали легкость шифрования.

Современная криптография основана на понятии односторонней функции f(x). Не вдаваясь в формальные математические определения, отметим одно ее свойство: инвертировать функцию, т. е. вычислить x, зная только f(x), крайне сложно. Стойкость шифров, помимо собственно алгоритма шифрования, во многом определяется и длиной ключа. Современная криптография исходит из того, что сам алгоритм рано или поздно все равно станет известен противнику. Все сообщения, передаваемые по открытым каналам связи, могут быть перехвачены, так что ключ шифра остается его единственным секретом.

Можно сказать, что толчком для развития теории информации в современном понимании стала работа Огюста Кергоффса «Военная криптография», опубликованная в 1883 г. Позднее Клод Шеннон в своей работе «Теория связи в секретных системах», опубликованной в 1949 г. [1], сформулировал необходимые и достаточные условия нерасшифруемости системы шифрования. Долгое время криптография оставалась секретной наукой, в тайны которой был посвящен лишь узкий круг лиц. Это было естественно, так как в первую очередь она была направлена на сохранение государственных секретов. Ситуация стала меняться во второй половине XX в. с появлением персональных компьютеров. Когда практически каждый человек получил возможность оперировать электронной информацией, возникла естественная потребность как-то защищать эту информацию от посторонних глаз. Широкое распространение получило использование симметричной криптографии, а несколько позднее и асимметричной. Также важную роль в современной криптографии играют поточные шифры и функции хэши-

рования. Данный курс посвящен именно этим четырем основным разделам криптографии.

Симметричное шифрование, которое в литературе еще называют традиционным шифрованием, или шифрованием с одним ключом, до изобретения шифрования с открытым ключом было единственным методом шифрования. Шифрование с секретным ключом впервые было описано в открытой литературе в 1976 г., когда в США был утвержден стандарт шифрования данных DES (Data Encryption Standard) [2]. Этот стандарт использовался довольно длительное время (более 20 лет), пока в 2001 г. не был принят новый стандарт AES (Advanced Encryption Standard), в основу которого лег алгоритм шифрования Rijndael.

Для симметричных алгоритмов шифрования характерны следующие свойства:

- использование одного и того же алгоритма как для зашифрования, так и для расшифрования данных;
- использование одного ключа, который является секретным;
- шифрование информации небольшими порциями (блоками); как правило, размер блока кратен 32 битам и составляет 64, 128, 192 или 256 битов.

К современным алгоритмам симметричного шифрования относятся такие шифры, как DES, AES (Rijndael), RC5, ГОСТ 28147-89 и многие другие.

В настоящий момент выделяют два основных способа построения симметричных алгоритмов шифрования: схему Фейстеля и сеть на основе подстановок и перестановок (SPN – Substitution-Permutation Network). По схеме Фейстеля построены алгоритмы DES, RC5, ГОСТ 28147-89 и др. Самым ярким представителем использования сети SPN является стандарт AES.

Традиционно считается, что концепция асимметричной криптографии впервые была предложена в 1976 г. Уитфилдом Диффи (Whitfield Diffie) и Мартином Хеллманом (Martin Hellman) и опубликована в том же году в основополагающей работе "Новые направления в криптографии" ("New Directions in Cryptography"). К числу отцов-основателей асимметричной крипто-

графии относят также и Ральфа Меркля (Ralph Merkle), который независимо от Диффи и Хеллмана пришел к тем же конструкциям, однако опубликовал свои результаты только в 1978 г.

С 1976 г. было создано множество криптографических алгоритмов, использующих концепцию открытых ключей. Многие из них не являются стойкими, а многие стойкие алгоритмы очень часто не пригодны для практической реализации, поскольку в них используется слишком большой ключ либо размер полученного с их помощью шифр-текста намного превышает объем открытого текста. И только весьма небольшая часть указанных алгоритмов являются и стойкими, и пригодными для практического использования. Как правило, эти алгоритмы основываются на решении одной из трудных математических задач, таких как задача дискретного логарифмирования или задача факторизации больших чисел [2]. Известны всего лишь три алгоритма, которые предоставляют достаточные возможности, как для шифрования текста, так и для его цифровой подписи: RSA, Эль-Гамаля и Рабина. Однако все эти алгоритмы работают достаточно медленно, зашифровывая и расшифровывая данные значительно медленнее, чем симметричные алгоритмы. В результате они часто непригодны для шифрования больших объемов данных, а используются для пересылки короткой шифрованной информации. Например, секретного ключа шифрования для симметричных криптосистем.

Для асимметричных алгоритмов шифрования характерны следующие свойства:

- не обязательно использование одного и того же алгоритма как для зашифрования, так и для расшифрования данных;
- использование двух взаимосвязанных ключей, один из которых является открытым, а второй – секретным.

По объему обрабатываемой информации все алгоритмы шифрования разделяют на блочные и поточные. Все упомянутые выше шифры относятся к блочным шифрам. Поточные шифры оперируют с битами (реже с байтами), стараясь обеспечить шифрование в режиме реального времени или близком к нему. Высокая скорость работы поточных шифров определяет

область их использования – закрытие данных, требующих оперативной доставки потребителю.

На сегодняшний день имеется достаточно большое число различных поточных шифров. Только в книге «Поточные шифры», вышедшей в 2003 г. описано более 20 шифров [3].

Одним из ярких представителей поточных шифров является шифр A5/1, который используется для шифрования связи GSM (Group Special Mobile – мобильная групповая специальная связь). Это европейский стандарт для мобильных цифровых сотовых телефонов. Он используется для шифрования канала «телефон/базовая станция».

Довольно большая группа поточных шифров основана на регистрах сдвига с линейной обратной связью (РСЛОС). Не исключение и шифр A5, в основе которого лежат сразу три РСЛОС.

Помимо алгоритмов шифрования, криптография охватывает также область разработки и применения функций хэширования. Криптографической функцией хэширования (хэш-функцией) H называется отображение множества всех возможных сообщений (представленных в двоичном виде) во множество двоичных векторов конечной фиксированной длины n (множество хэш-значений или хэш-кодов) [4].

Современные хэш-функции используются в таких задачах, как [5]:

- проверка целостности файлов, архивов, сборок;
- хранение паролей в необратимом виде (в хранилище помещаются не сами пароли, а хэш-суммы от строки, что позволяет при раскрытии хэша сохранить собственно словарную парольную фразу в секрете);
- электронно-цифровая подпись документов.

Помимо вышеперечисленных задач, которые решаются с использованием функций хэширования, на их основе иногда строятся следующие криптографические примитивы: блочные шифры (например, Shacal-2), поточные шифры, генераторы псевдослучайных чисел [4].

В 1989 г. Р. Меркль (Ralph C. Merkle) и И. Дамгорд (Ivan Damgaard) [4] независимо предложили итеративный принцип

построения криптографических функций хэширования. Данный принцип позволяет свести задачу построения хэш-функции на множестве сообщений различной длины к задаче построения отображения, действующего на множестве фиксированной конечной длины. По итеративному принципу построено абсолютное большинство хэш-функций, используемых в настоящее время на практике. Например, хэш-функции MD5, SHA-1, семейство хэш-функций SHA-2, отечественный стандарт на хэш-функцию ГОСТ Р 34.11-94.

Настоящее учебное пособие посвящено изучению основных аспектов современной криптографии, а именно ее большому разделу – симметричным блочным шифрам. Большое количество наглядных примеров позволит освоить основные принципы применения криптографических алгоритмов. А вопросы для самоконтроля и задачи для самостоятельного решения будут способствовать закреплению изученного материала. Учебное пособие рассчитано на студентов, магистрантов и аспирантов направлений 10.03.01 «Информационная безопасность», 10.05.02 «Информационная безопасность телекоммуникационных систем», 10.05.03 «Информационная безопасность автоматизированных систем». Изученный материал позволит самостоятельно применять полученные знания на практике как в области криптографии, так и в области криптоанализа.

1. ОСНОВНЫЕ ОПЕРАЦИИ, ИСПОЛЬЗУЕМЫЕ В СОВРЕМЕННЫХ КРИПТОГРАФИЧЕСКИХ СИСТЕМАХ

Любой алгоритм шифрования – это совокупность некоторых математических преобразований, которые циклически повторяются некоторое количество раз. Несмотря на такое большое количество различных алгоритмов шифрования, на самом деле количество базовых криптографических примитивов не так уж велико. Все дело в том, что каждый примитив (каждая операция) может иметь свою интерпретацию, свои значения, свои какие-то данные. Именно этим фактом, а также тем, что скомбинировать операции можно по-разному, объясняется такое большое количество шифров. Конечно, нижеперечисленные операции не охватывают всю область построения современных криптосистем. Существуют другие, более сложные и специфические преобразования. Некоторые из них были придуманы для конкретных шифров, как, например, операция перемешивания MixColumns для алгоритма шифрования Rijndael. Те операции, которые мы рассмотрим ниже, имеют широкое применение и используются в большинстве известных алгоритмов. Поэтому прежде чем перейти непосредственно к изучению различных схем шифрования данных, давайте разберемся с тем, как работают их составные части по отдельности.

Сложение по модулю 2 (исключающее ИЛИ, операция XOR)

Операция сложения по модулю 2 работает следующим образом: если складываются два одинаковых бита (т. е. или два нуля, или две единицы), то в результате сложения образуется 0; если же складываются два разных бита (0 и 1 или 1 и 0), то в результате сложения образуется 1:

$$a \oplus b = \begin{cases} 0, если\ a = b, \\ 1, если\ a = b. \end{cases}$$

Пример. Пусть даны два четырехбитовых числа a = 0110, b = 1101. Тогда

$$\begin{array}{r} \oplus \begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{matrix} \\ \hline \begin{matrix} 1 & 0 & 1 & 1 \end{matrix} \end{array}$$

Целочисленное сложение по модулю 2^n

Результат сложения по модулю 2^n не должен превышать значение модуля. Если же при сложении сумма оказалась больше чем 2^n, то ее нужно скорректировать путем вычитания из нее значения 2^n:

$$(a + b) \bmod 2^n = \begin{cases} a + b, если\ (a + b) < 2^n, \\ a + b - 2^n, если\ (a + b) \geq 2^n. \end{cases}$$

Пример.

$(5 + 7) \bmod 2^4 = 12 \bmod 2^4 = 12.$

$(12 + 11) \bmod 2^4 = 23 \bmod 2^4 = 7.$

Целочисленное умножение по модулю 2^n

Результат умножения по модулю 2^n не должен превышать значение модуля. Если же при умножении произведение оказалось больше чем 2^n, то в этом случае результатом произведения по модулю будет являться остаток от деления найденного произведения на значение модуля 2^n:

$$ab \bmod 2^n = \begin{cases} ab, если ab < 2^n, \\ \left\lceil \frac{ab}{2^n} \right\rceil, если ab \geq 2^n. \end{cases}$$

Пример.

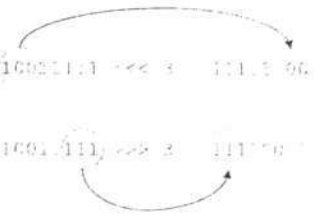
$4*3 \bmod 2^4 = 12 \bmod 2^4 = 12.$

$3*12 \bmod 2^4 = 36 \bmod 2^4 = 4.$

Циклический сдвиг на заданное число позиций

Операция циклического сдвига влево обозначается знаком <<<; а операция циклического сдвига вправо – знаком >>>. Для циклического сдвига отличительной особенностью является тот факт, что биты, вытолкнутые за разрядную сетку в результате сдвига, помещаются в противоположный конец регистра сдвига на освободившиеся позиции.

Пример.



Таблицы перестановок

Различают несколько видов для таблиц перестановок:

- простые перестановки;
- перестановки со сжатием;
- перестановки с расширением.

Простые перестановки меняют биты в двоичной последовательности местами. Перестановки со сжатием меняют биты исходной последовательности местами, но при этом используют не все биты исходной последовательности. Перестановки с расширением также меняют биты исходной позиции местами, но при этом некоторые биты исходной последовательности используются дважды, за счет чего результирующая последовательность битов оказывается большей длины.

Пример. Использование простой перестановки. Пусть простая перестановка задана табл. 1.

Таблица 1

Таблица перестановки

8	7	3	2	5	4	1	6

На вход преобразования поступает последовательность 10100011. Для того чтобы применить перестановку, необходимо пронумеровать биты исходной последовательности от 1 до 8 слева направо. После этого биты исходной последовательности необходимо преобразовать с помощью табл. 1. Так, в соответствии с табл. 1, восьмой бит исходной последовательности станет первым, седьмой бит – вторым, третий бит останется треть-

им и т.д. В результате преобразования исходная последовательность 1010 0011 приводится к виду 1110 0010. Для наглядности процесс применения перестановки с расширением изображен на рис. 1.

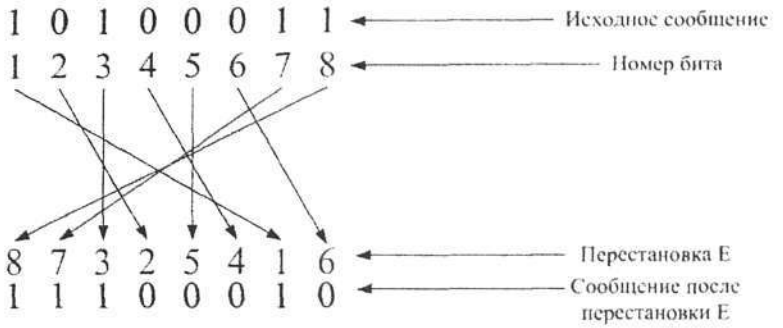


Рис. 1. Выполнение простой перестановки

Пример.

Использование перестановки с расширением. Пусть простая перестановка задана в табл. 2.

Таблица 2

Перестановка с расширением

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

На вход преобразования поступает последовательность 10100011. Для того чтобы применить перестановку, необходимо пронумеровать биты исходной последовательности от 1 до 8 слева направо. После этого биты исходной последовательности необходимо преобразовать с помощью табл. 2. Так, в соответствии с табл. 2, третий бит исходной последовательности станет первым, четвертый бит – вторым, первый бит – третьим и т.д. В результате преобразования исходная последовательность 1010 0011 приводится к виду 1010 0101 1100. Биты исходной последовательности под номерами 2, 3, 4 и 8 дублируются и дважды встречаются в результирующей строке. Для наглядности про-

цесс применения перестановки с расширением изображен на рис. 2.

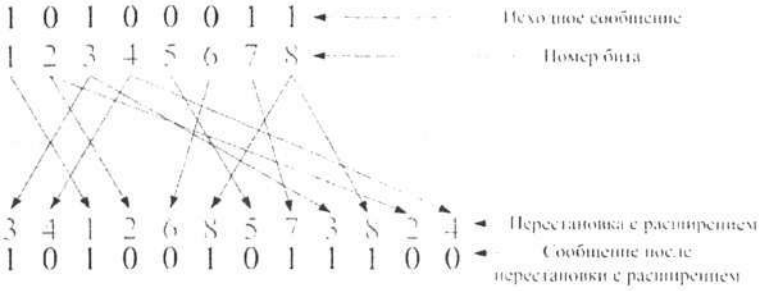


Рис. 2. Выполнение перестановки с расширением

Пример.

Использование перестановки со сжатием. Пусть простая перестановка задана в табл. 3.

Таблица 3

Перестановка со сжатием

12	3	9	7	1	10	8	4

На вход преобразования поступает последовательность из 12 битов 101000110110. Для того чтобы применить перестановку, необходимо пронумеровать биты исходной последовательности от 1 до 12 слева направо. После этого биты исходной последовательности необходимо преобразовать с помощью табл. 3. При этом в перестановке будут задействованы не все биты исходной последовательности, а только 8 битов из 12 (отсутствуют биты 2, 5, 6, 11). Таким образом, в результате перестановки будет образована последовательность меньшей длины. Таблицы перестановки со сжатием используются, например, в процедурах выработки раундовых подключей, когда из секретного ключа необходимо выработать подключи меньшего размера. Для наглядности процесс применения перестановки с расширением изображен на рис. 3.

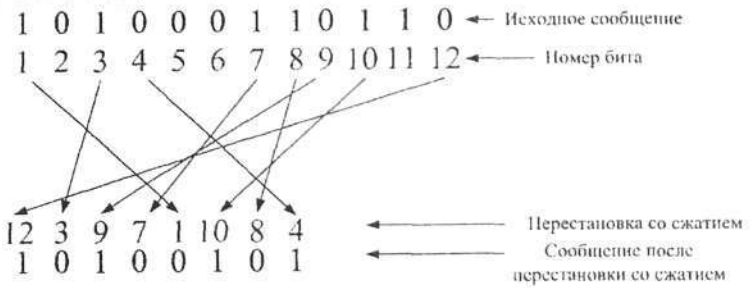


Рис. 3. Выполнение перестановки со сжатием

Блоки замены

Блоки замены используются для замены одной последовательности битов на другую в соответствии с некоторыми таблицами данных. При этом размеры исходного и итогового сообщений могут разниться. Рассмотрим два варианта использования таблиц для изменения значения сообщения с помощью блока замены.

Вариант 1. Исходное сообщение длиной 4 бита преобразовывается после прохождения через блок замены к сообщению из 3 битов. Таблица замены имеет две строки и восемь столбцов и представлена в табл. 4.

Пусть на вход блока замены поступает значение 1011. Первый бит последовательности указывает на номер строки в табл. 4, последние три бита – на номер столбца. В результате такого преобразования на выходе блока замены образуется значение 100. Для наглядности процесс преобразования данных с помощью первого блока замены представлен на рис. 4.

Таблица 4

Таблица для первого варианта блока замены

a2a3a4 / **a1**	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0	4	6	1	3	5	7	2	5
1	5	7	2	4	6	1	3	6

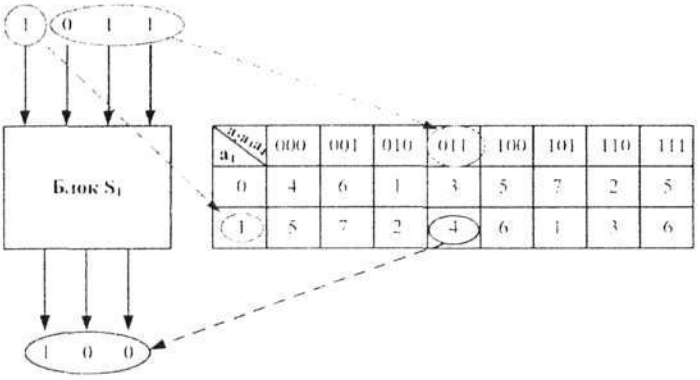


Рис. 4. Первый вариант замены с помощью S-блока

Вариант 2. Исходное сообщение длиной 4 бита преобразовывается после прохождения через блок замены к сообщению из 2 битов. Таблица замены имеет 4 строки и 4 столбца (табл. 5).

Пусть на вход блока замены поступает значение 0001. Первый и последний биты входа в блок замены, т. е. значение 01, образуют номер строки в таблице замены, а средние два бита, т. е. значение 00, образуют номер столбца. В результате преобразования на выходе блока замены образуется значение 10. Для наглядности процесс преобразования данных с использованием последнего блока замены представлен на рис. 5.

Таблица 5

Таблица для второго варианта блока замены

a2a3 / **a1 a4**	**00**	**01**	**10**	**11**
00	1	3	2	1
01	2	1	3	2
10	3	2	1	3
11	1	3	2	1

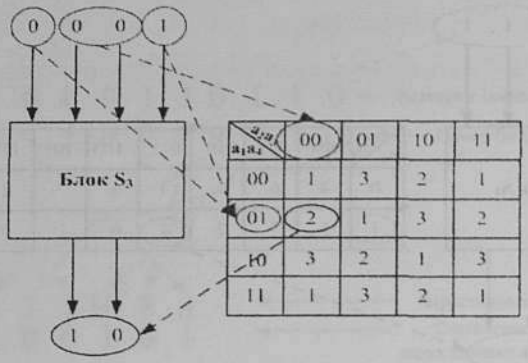


Рис. 5. Второй вариант замены данных с помощью S-блока

Выполнение обратных перестановок

Простые перестановки и перестановки с расширением, представленные выше, являются обратимыми. Это свойство широко используется в криптоанализе современных шифров. Рассмотрим более подробно, как можно обратить последовательность битов, т. е. восстановить порядок битов, который был в сообщении до применения к нему перестановки.

Исходная перестановка с расширением предназначена для перестановки исходной последовательности битов в соответствии с некоторым правилом. В рассматриваемом нами случае в качестве такого правила выступает табл. 2, в соответствии с которой третий бит исходной позиции должен стать первым, четвертый бит – вторым и т.д. При этом отличительной особенностью работы перестановки с расширением от других перестановок является дублирование некоторых битов исходной последовательности. Так, в соответствии с табл. 2, третий бит исходной позиции станет не только первым битом новой последовательности, но еще будет продублирован в девятой позиции. Задача обратной перестановки сводится к восстановлению исходной последовательности битов, т. е. последовательности битов, которая поступала на вход перестановки с расширением E (от англ. expansion). Так, нам известно, что первый бит исходной последовательности был помещен в третью позицию. Таким образом, чтобы восстановить значение бита, надо третий

бит результата перестановки отправить на первое место. Аналогичным образом видно, что второй бит исходной последовательности был помещен в четвертую позицию, поэтому для того чтобы восстановить значение бита, надо четвертый бит результата перестановки отправить на второе место. Для наглядности процесс восстановления таблицы для работы обратной перестановки с расширением E^{-1} представлен на рис. 6.

Рис. 6. Пример выполнения обратной перестановки

Из рис. 6 видно, что последние четыре бита перестановки E, которые являются дублированными битами исходной последовательности, остаются незадействованными, так как ранее они уже были использованы. Это логично, так как нам известно, что на вход перестановки с расширением поступает 8-битовая последовательность, которая преобразуется к 12-битовой последовательности. Таким образом, обратная перестановка E^{-1} должна восстанавливать 8 битов из 12, что и отражено на рис. 6.

Процесс обращения обычной перестановки P (от англ. *permutation*) аналогичен за тем исключением, что при этом отсутствуют дублирующие биты. Для наглядности процесс восстановления обратной перестановки P^{-1} представлен на рис. 7.

Рис. 7. Пример выполнения обратной перестановки

2. ОСНОВНЫЕ СХЕМЫ ПОСТРОЕНИЯ СИММЕТРИЧНЫХ АЛГОРИТМОВ ШИФРОВАНИЯ

Схема Фейстеля, или сеть Фейстеля (англ. *Feistel*, иногда в русскоязычной литературе можно встретить перевод Файстель), была первоначально предложена при разработке алгоритма шифрования Люцифер (Lucifer), который во многом схож со стандартом DES. Схема оказалась настолько удачной, что по такому принципу конструировались практически все блочные шифры до появления алгоритма Rijndael, разработчики которого предложили использовать другой способ построения блочных шифров.

Схема Фейстеля представляет собой разновидность итерированного блочного шифра. При зашифровании блок открытого текста разделяется на две равные части – правую и левую. При этом длина исходного блока данных должна быть четной. В каждом цикле одна из частей подвергается преобразованию при помощи функции F и подключа Ki, полученного из исходного секретного ключа K. Результат операции суммируется по модулю 2 (операция XOR) с другой частью. Затем левая и правая части меняются местами. Общий вид одного цикла алгоритма шифрования, построенного по схеме Фейстеля, представлен на рис. 8. Преобразования в каждом цикле одинаковы, но на последнем не выполняется перестановка. Процедура расшифрования аналогична процедуре зашифрования за тем исключением, что подключи ki выбираются в обратном порядке. Если же при зашифровании в последнем цикле была выполнена перестановка, то расшифрование следует начать с перестановки левой и правой частей блока данных.

Уже отмечалось, что к алгоритмам шифрования, построенным по схеме Фейстеля, относятся такие алгоритмы, как DES и ГОСТ 28147-89. Помимо вышеуказанных алгоритмов существует целый ряд други, таких как, например, Lucifer, FEAL, Khufu, Khafre, LOKI, Blowfish, также построенных по схеме Фейстеля. Блочный алгоритм шифрования, использующий описанную конструкцию, является обратимым и гарантирует воз-

можность восстановления входных данных функции F в каждом цикле. Сама функция F не обязательно должна быть обратимой. При задании произвольной функции F не потребуется реализовывать две различные процедуры – одну для шифрования, а другую для расшифрования. Ниже мы рассмотрим почему это возможно.

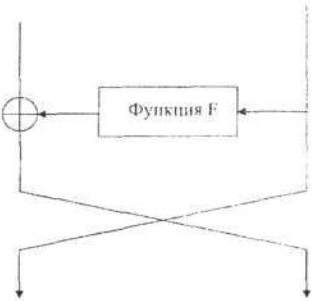


Рис. 8. Один раунд алгоритма шифрования, построенный по схеме Фейстеля

Необходимо также отметить, что деление исходного шифра на две части может быть заменено делением на четыре, восемь частей и более. Такие алгоритмы шифрования называются производными от схемы Фейстеля. Некоторые современные шифры имеют такую структуру. Например, алгоритм шифрования CAST или закрытый в свое время алгоритм шифрования SkipJack. На рис. 9 показаны пример некоторых модификаций схемы Фейстеля.

Разработчики алгоритма Rijndael ушли от общепринятой схемы Фейстеля и организовали свой шифр так, чтобы не разделять шифруемый текст на половинки, а выполнять все преобразования над цельным блоком данных. Такая структура получила название SPN или SP-сеть или Сеть на основе подстановок и перестановок (от англ. *substitution-permutation network*). По такому же принципу были построены и некоторые другие алгоритмы шифрования, представленные в качестве кандидатов на конкурс AES, например, алгоритмы шифрования MARS и Ser-

pent. В качестве примера организации шифрования с помощью SP-сети приведем алгоритм шифрования Threefish, лежащий в основе функции хэширования Skein (рис. 10).

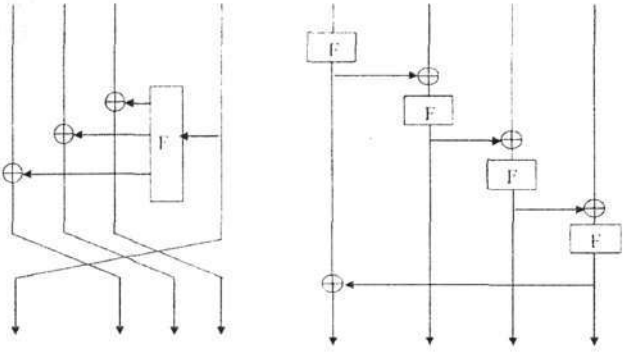


Рис. 9. Модификация схемы Фейстеля

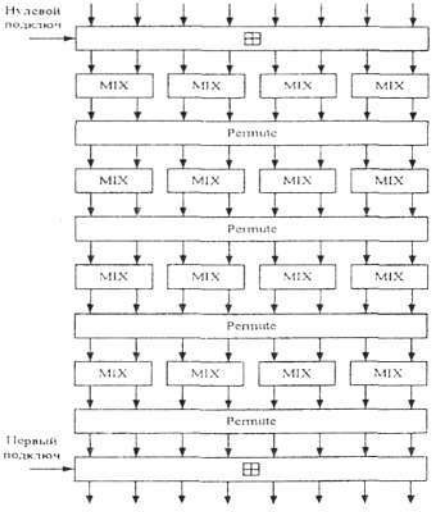


Рис. 10. Пример построения шифра на основе SP-сети

3. СТАНДАРТЫ ШИФРОВАНИЯ ДАННЫХ

3.1. Бывший стандарт шифрования данных США – DES

Несмотря на то, что с мая 2002 г. в США в силу вступил новый стандарт шифрования данных AES, предыдущий стандарт шифрования данных DES (Data Encryption Standard), который ANSI называет алгоритмом шифрования данных DEA (Data Encryption Algorithm), а ISO – DEA-1, остается одним из самых известных алгоритмов. Можно даже сказать, что в настоящий момент алгоритм шифрования DES используется в основном для обучения специалистов по защите информации, являясь открытым, и в то же время стойким алгоритмом шифрования. В том числе до сих пор широко используется схема тройного алгоритма DES с двумя ключами. За годы своего существования алгоритм выдержал натиск различных атак и утратил свою былую силу лишь потому, что вычислительная мощность за 25 лет использования DES в качестве стандарта выросла во много раз. Так, если во время разработки шифра перебор всех вариантов секретного ключа занял бы несколько сотен лет при использовании самого мощного компьютера того времени, то в настоящий момент на это требуются считанные часы при использовании самого мощного из существующих суперкомпьютеров.

DES представляет собой блочный симметричный шифр, построенный по схеме Фейстеля, и преобразует 64-битовый блок исходных данных в блок шифрованных данных такой же длины под воздействием секретного ключа шифрования, длина которого равна 56 бит.

Так как DES является симметричным алгоритмом, то для шифрования и расшифрования используются одинаковые алгоритмы с той лишь разницей, что при расшифровании раундовые подключи используются в обратном порядке. То есть, если при шифровании использовались раундовые подключи K_1, K_2, K_3,... K_{16}, то подключами расшифрования соответственно будут K_{16}, K_{15}, K_{14},... K_1. Алгоритм использует только стандартную арифметику 64-битовых чисел и логические операции, поэтому легко реализуется на аппаратном уровне.

Длина секретного ключа равна 56 битам. Ключ обычно представляется 64-битным числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа.

Фундаментальным строительным блоком DES является комбинация подстановок и перестановок. DES работает с 64-битовыми блоками открытого текста и состоит из 16 раундов. Работа алгоритма начинается с первоначальной перестановки, выполняемой до начала работы раундов шифра в соответствии с табл. 6. В оригинале описания данная перестановка названа IP (от англ. *Initial Permutation*).

Эту и все последующие таблицы для алгоритма DES следует читать слева направо и сверху вниз. Например, начальная перестановка перемещает бит 58 в позицию 1, бит 50 – в позицию 2, бит 42 – в позицию 3 и так далее.

Таблица 6

Начальная перестановка

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

После этого блок данных разбивается на правую и левую половины длиной по 32 бита. Затем выполняется 16 раундов, в которых данные перемешиваются и объединяются с ключом. После шестнадцатого раунда правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной). Общий вид работы алгоритма представлен на рис. 11.

В каждом раунде правая половина данных увеличивается с 32 до 48 битов путем дублирования некоторых битов с помощью перестановки с расширением в соответствии с табл. 7.

Расширенные данные объединяются с 48-битным раундовым подключом посредством сложения по модулю 2 (операция XOR), после чего проходит через восемь S-блоков замены. На вход каждого S-блока поступает 6-битный вход и в результате замены образуется 4-битный выход. Всего используется восемь

различных S-блоков. Каждый S-блок представляет собой таблицу из четырех строк и шестнадцати столбцов. Каждый элемент в блоке является 4-битным столбцом. По шести входным битам S-блока определяется, под какими номерами столбцов и строк следует искать выходное значение. Первый из восьми S-блоков показан в табл. 8. Остальные блоки замены легко можно найти в книгах и справочниках по криптографии, например, в работах [2, 6 – 9].

Таблица 7

Перестановка с расширением

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

Таблица 8

Блок S_1

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Входные биты особым образом определяют элемент S-блока. Рассмотрим 6-битовый вход S-блока: b_1, b_2, b_3, b_4, b_5 и b_6. Биты b_1 и b_6 объединяются, образуя 2-битное число от 0 до 3, соответствующее строке таблицы. Средние четыре бита, с b_2 по b_5, объединяются, образуя 4-битное число от 0 до15, соответствующее столбцу таблицы. Необходимо учитывать, что строки и столбцы нумеруются с нуля, а не с единицы. Например, пусть на вход блока S_1 попадает значение 110011. Первый и последний биты, объединяясь, образуют 11, что соответствует строке 3 блока S_1. Средние 4 бита образуют 1001, что соответствует столбцу 9 того же S-блока. Элемент блока S_1, находящийся на пересечении строки 3 и столбца 9, – это 11, т. е. на выходе будет образовано значение 1011 (рис. 12).

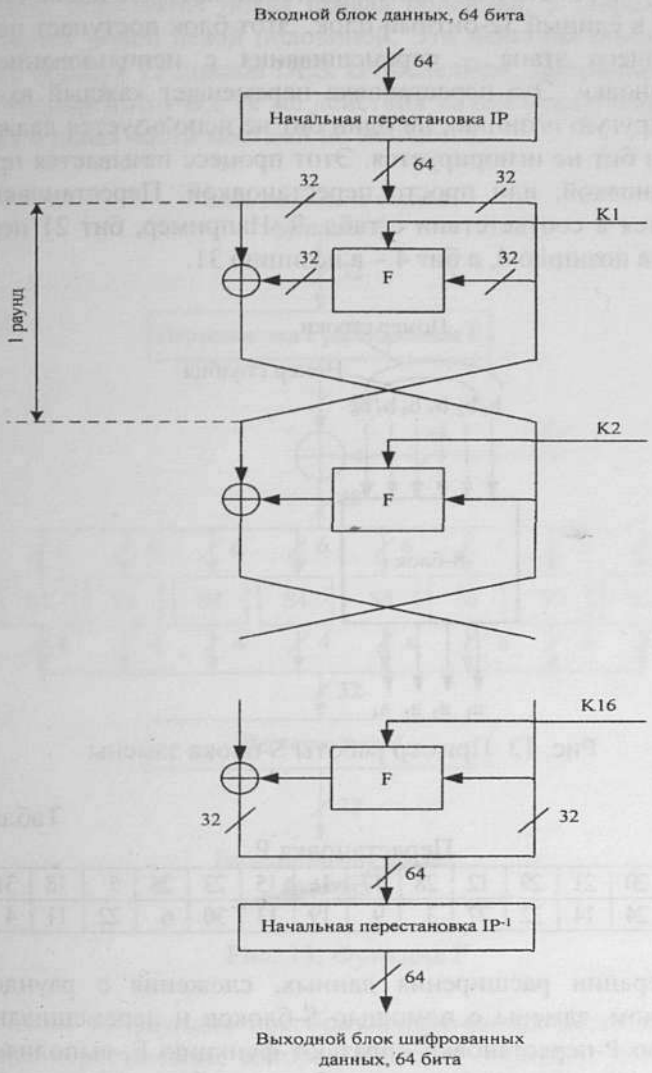


Рис. 11. Общий вид работы алгоритма шифрования DES

В результате подстановки с помощью восьми блоков замены получается восемь 4-битных блоков, которые вновь объединяются в единый 32-битный блок. Этот блок поступает на вход следующего этапа – перемешивания с использованием P-перестановки. Эта перестановка перемещает каждый входной бит в другую позицию, ни один бит не используется дважды, и ни один бит не игнорируется. Этот процесс называется прямой перестановкой, или просто перестановкой. Перестановка выполняется в соответствии с табл. 9. Например, бит 21 перемещается в позицию 4, а бит 4 – в позицию 31.

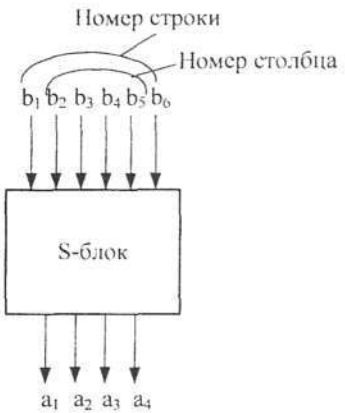


Рис. 12. Пример работы S-блока замены

Таблица 9

Перестановка P

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

Операции расширения данных, сложения с раундовым подключом, замены с помощью S-блоков и перемешивания с помощью P-перестановки образуют функцию F, выполняемую в каждом раунде шифрования. Схема работы функции F наглядно показана на рис. 13.

Результат функции F объединяется с левой половиной с помощью операции сложения по модулю 2 (XOR). В итоге этих

действий появляется новая правая половина, а старая правая становится новой левой половиной. Эти действия повторяются 15 раз, образуя 15 циклов DES. В последнем, шестнадцатом раунде выполняются все те же действия за тем исключением, что правая и левая части местами не меняются.

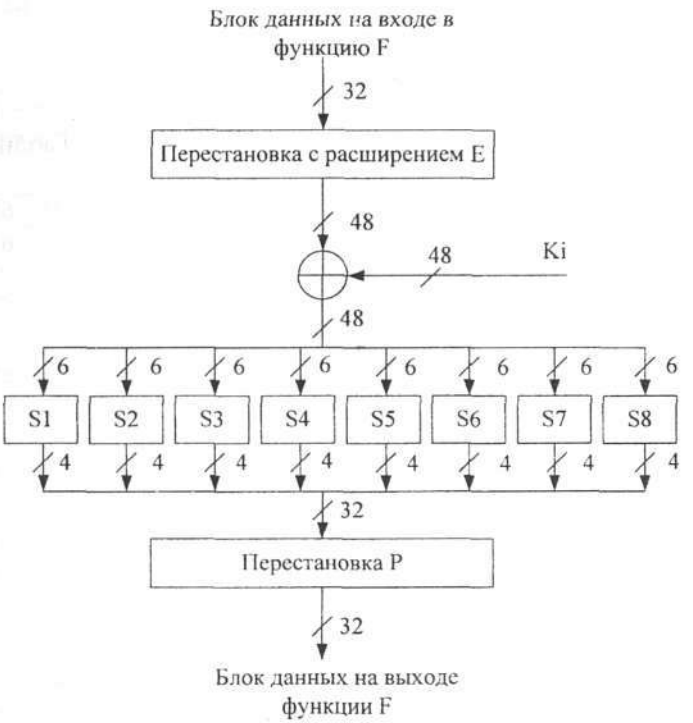


Рис. 13. Функция F

После 16 раундов преобразования выполняется заключительная перестановка, которая является обратной по отношению к первоначальной и преобразует данные в соответствии с табл. 10. В оригинале описания данная перестановка обозначена как IP^{-1}.

Начальная перестановка IP и заключительная перестановка IP^{-1} связаны между собой. Говорят, что они взаимно-обратны.

Это означает, что если вы к некоторому сообщению X примените сначала одну перестановку, а затем вторую, то в результате такого действия вы получите исходное сообщение X. И при этом не важно, какую из перестановок вы будете применять первой, а какую последней. То есть:

$$IP^{-1}(IP(X)) = X;$$
$$IP(IP(X^{-1})) = X.$$

Таблица 10

Заключительная перестановка

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

И действительно, если начальная перестановка (см. табл. 6) переставляет бит 58 в позицию 1, бит 50 в позицию 1, бит 42 – в позицию 3; то заключительная перестановка (табл. 10) возвращает их на места, переставляя бит 1 в позицию 58, бит 2 в позицию 50, бит 3 в позицию 42 и т.д.

Остается рассмотреть процедуру выработки шестнадцати 48-битовых раундовых подключей из секретного 56-битного ключа шифрования. Она сводится к следующим действиям. Сначала 64-битовый ключ DES уменьшается до 56-битового ключа отбрасыванием каждого восьмого бита, как показано в табл. 11. При этом не только происходит отброс битов четности, но и непосредственно перестановка битов.

Таблица 11

Начальное преобразование ключа

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

После извлечения 56-битного ключа для каждого раунда шифрования генерируется новый 48-битный подключ K_i следующим образом. Сначала 56-битный ключ делится на две 28-битные половины. Затем половины циклически сдвигаются влево на один или два бита в зависимости от номера раунда. Зависимость величины сдвига от номера раунда показана в табл. 12.

Таблица 12

Сдвиг ключа при выработке подключей для шифрования

№ раунда	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг t (бит)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

После сдвига выбирается 48 битов из 56. Так как при этом не только выбирается подмножество битов, но и изменяется их порядок, эта операция получила название перестановка со сжатием. Ее результатом является набор из 48 бит. Перестановка со сжатием выполняется в соответствии с табл. 13.

Таблица 13

Перестановка со сжатием

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

Например, бит сдвинутого ключа в позиции 33 перемещается в позицию 35 результата, а 18-й бит сдвинутого ключа отбрасывается.

Схема выработки раундовых подключей показана на рис. 14.

Из-за сдвига для каждого подключа используются различные подмножества битов ключа. Каждый бит используется приблизительно в четырнадцати из шестнадцати подключей, при этом не все биты используются одинаковое число раз.

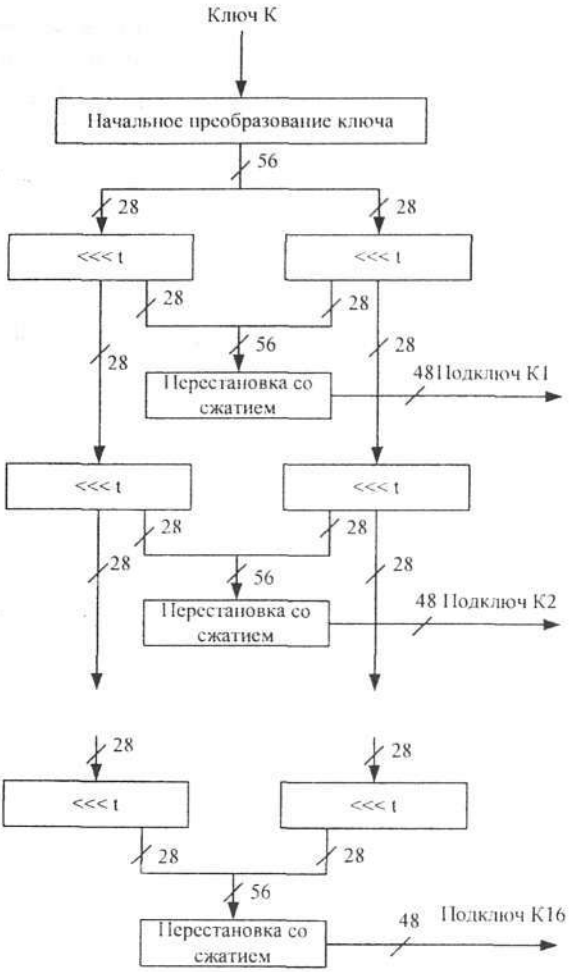


Рис. 11. Схема выработки раундовых подключей

При расшифровании нужно использовать подключи в обратном порядке. Для их последовательного извлечения достаточно изменить алгоритм выработки подключей следующим образом: производить циклический сдвиг вправо в соответствии с табл. 14.

Таблица 14

Сдвиг ключа при выработке подключей для расшифрования

№ раунда	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг (бит)	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Это интересно

Для алгоритма DES существует ряд слабых и условно слабых подключей. Слабых ключей четыре. Это такие ключи, которые позволяют выработать 16 абсолютно идентичных раундовых подключа. Если представить ключ в 16-ричном виде, то эти 4 слабых ключа будут выглядеть следующим образом:

0101 0101 0101 0101
1F1F 1F1F 1F1F 1F1F
E0E0 E0E0 E0E0 E0E0
FEFE FEFE FEFE FEFE

После первоначальной перестановки данные ключи разделятся на две 28-битовых половинки, каждая из которых будет выглядеть следующим образом соответственно для каждого из ключей:

0000000 0000000
0000000 FFFFFFF
FFFFFFF 0000000
FFFFFFFFFFFFFF

Невооруженным глазом видно, что каждая половинка состоит либо из нулей, либо из единиц. Поэтому после сдвига половинки останутся неизменными, и для каждого из 16 раундов на вход перестановки со сжатием будет поступать одно и то же значение, которое будет приводить к выработке одного и того же раундового подключа.

Помимо четырех слабых ключей для алгоритма DES выделяют так называемые полуслабые ключи. Полуслабые ключи имеют парный ключ, который позволяет выполнить над текстом точно такие же преобразования. То есть сообщение, зашифрованное одним ключом, легко может быть расшифровано другим ключом. Известно шесть пар таких ключей.

Также существуют возможно слабые ключи. Эти ключи вырабатывают всего четыре раундовых подключа, которые циклически используются четыре раза, что значительно ослабляет стойкость шифра. Таких ключей насчитывают 48 штук.

3.2. Стандарт шифрования ГОСТ 28147-89

Алгоритм шифрования ГОСТ является стандартом шифрования данных в Российской Федерации. Это означает, что он является обязательным для применения в качестве алгоритма шифрования во всех государственных организациях РФ. На его основе построено большинство безопасных информационных систем (Secure Information Systems), например, таких как CryptoProCSP, VipNetCSP, LissiCSP, Secret Disk и многие другие. Помимо этого, алгоритм ГОСТ используется в составе функции хэширования и входит в стандарт RFC4357 в части "id-GostR3411-94-CryptoProParamSet".

Впервые широкая общественность узнала о данном шифре в 1994 г., когда он был рассекречен и переведен на английский язык. Несмотря на то, что стандарт был разработан более 20 лет назад, в 2010 г. рассматривалась возможность принятия алгоритма ГОСТ в качестве международного стандарта шифрования в ISO 18033.

Стандарт шифрования ГОСТ 28147-89 (далее по тексту мы будем опускать цифры и использовать сокращенное название ГОСТ) имеет четыре основных режима работы: режим простой замены, режим гаммирования, режим гаммирования с обратной связью и режим выработки имитовставки. Все режимы используют одно и то же основное преобразование, но с разным числом раундов и различным образом.

В основе всех режимов лежит работа алгоритма ГОСТ в режиме простой замены. Именно с него мы и начнем наше рассмотрение.

В описании стандарта программно-аппаратная реализация алгоритма ГОСТ схематично представлена так, как показано на рис. 15.

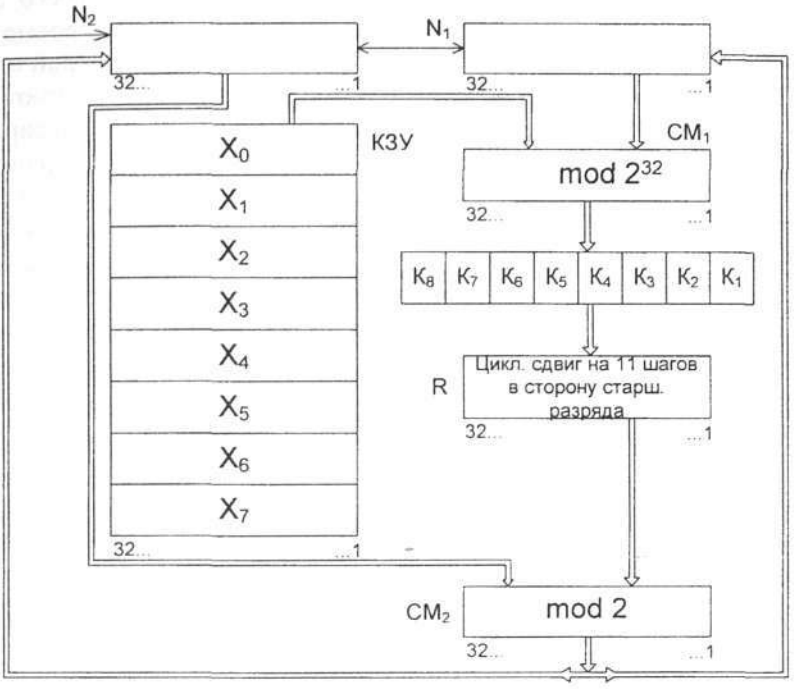


Рис. 15. Программно-аппаратная схема алгоритма ГОСТ в режиме простой замены

Прежде чем перейти к детальному рассмотрению рис. 15, мы предлагаем попробовать рассмотреть алгоритм ГОСТ в привычном «классическом» представлении схемы Фейстеля. Алгоритм шифрования ГОСТ 28147-89 в режиме простой замены представляет собой блочный алгоритм шифрования, построенный по схеме Фейстеля, с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками (рис. 16).

Для шифрования открытый текст разбивается на две равные части по 32 бита каждая. В i-м цикле используется подключ K_i. Функция F реализована следующим образом. Сначала правая половина данных и подключ K_i складываются по модулю 2^{32}. Результат разбивается на восемь 4-битовых последова-

тельностей, каждая из которых поступает на вход своего S-блока. ГОСТ использует восемь различных S-блоков, первые 4 бита попадают в первый S блок, вторые четыре – во второй и т. д. Каждый S-блок представляет собой перестановку чисел от 0 до 15. Все восемь S-блоков различны и не являются фиксированными. То есть при шифровании можно использовать произвольный набор из восьми перестановок в качестве восьми S-блоков замены. Выходы всех восьми S-блоков объединяются в 32-битное слово, которое циклически сдвигается влево на 11 битов. Наконец, результат объединяется с помощью операции сложения по модулю 2 (XOR) с левой половиной данных. В каждом раунде за исключением последнего правая и левая части меняются местами.

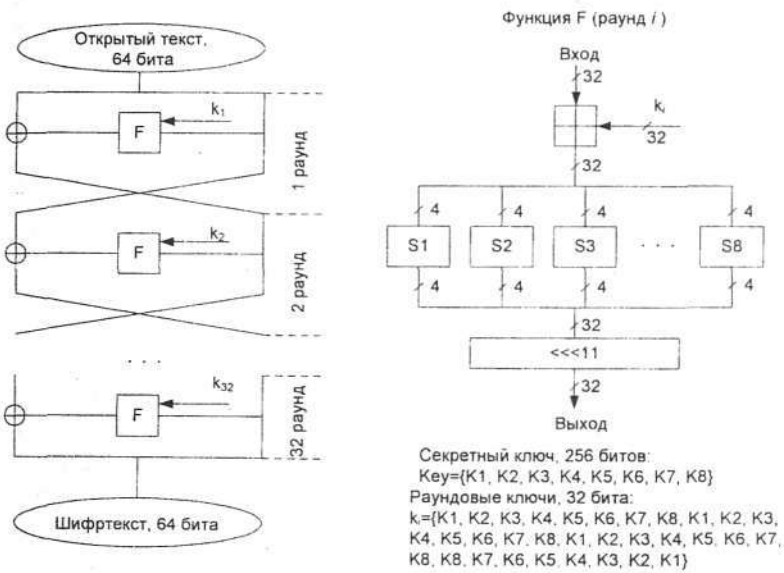


Рис.16. Алгоритм шифрования ГОСТ - 28147-89 в режиме простой замены

Исходный секретный ключ К имеет размер 256 битов и состоит из восьми 32-битовых слов: K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7.

Расширение материала ключа производится по схеме K[i]=(K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0). При расшифровании используется тот же самый алгоритм за тем исключением, что раундовые подключи используются в обратном порядке.

Алгоритм слабо подвержен распараллеливанию, а вот в плане многоплатформенности он имеет достаточно «удобный» дизайн.

Режим простой замены предназначен для шифрования ключей (существует множество схем применения алгоритмов симметричного шифрования, использующих несколько ключей различного назначения; в этих случаях требуется шифрование одних ключей на другие). В данном режиме выполняется 32 раунда основного преобразования. В каждом из раундов, как было сказано выше, используется определенный подключ, как было описано выше.

Теперь попробуем соотнести между собой рис. 15 и 16, для того чтобы разобраться с логикой работы программно-аппаратной схемы. Для этого обратимся к рис. 17.

Итак, шифруемый текст подается в накопители N1 и N2 (на рис. 17 – это связь №1). Накопители 32-разрядные. Сначала первые 32 бита текста заносятся в накопитель N1 и переписываются в N2. После этого вторые 32 бита текста записываются в накопитель N1. Таким образом получается, что в накопителе N1 находится левая половина блока шифруемого текста, а в накопителе N2 – правая. Дальше правая часть (содержимое накопителя N2) поступает на вход функции F (связь 2). После чего выполняется сложение по модулю 2 данных с очередным раундовым подключом (связь 3). Все раундовые подключи хранятся в КЗУ (связь 4) и в программно-аппаратной схеме обозначены буквой X. После этого происходит преобразование данных с помощью блоков замены (связь 5). В программно-аппаратной схеме стандарта блоки замены обозначены буквой K. Далее выполняется циклический сдвиг в сторону старших позиций на 11 разрядов (связь 6). Результат данной операции является выходом функции F. После этого результат работы функции F необходимо сложить с левой половиной блока данных. Поэтому со-

держимое регистра N1 поступает на вход сумматора СМ2 (связь 7), где и происходит поразрядное сложение данных (связь 8). Во всех раундах, кроме последнего, результат сложения выхода функции F с левой частью данных должен переместиться в правую часть. Поэтому сначала содержимое регистра N2 переписывается в регистр N1 (связь 10), после чего результат работы сумматора СМ2 переписывается в накопитель N2 (связь 9).

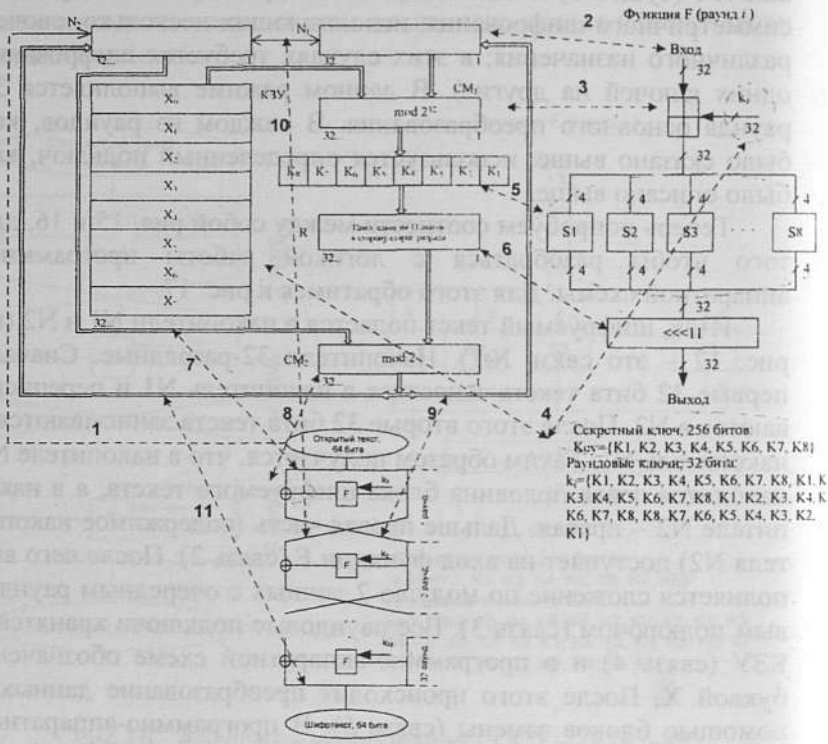


Рис. 17. Соотношение классического представления схемы Фейстеля с программно-аппаратным представлением алгоритма ГОСТ

В последнем раунде смена половинок блока данных отсутствует, поэтому содержимое регистра N2 остается неизменным, а результат работы сумматора СМ2 переписывается в накопитель N1 (связь 11).

Это интересно

Отличительной чертой алгоритма ГОСТ является использование в его структуре нефиксированных блоков замены. Предполагается, что при любом заполнении S-блоков тридцати двух раундов шифрования будет достаточно для того, чтобы противостоять любым методам криптоанализа. Долгое время считалось, что если оставлять S-блоки в секрете, то их можно рассматривать как дополнительный ключевой материал. Однако в работе [10] был предложен метод, применение которого позволяет достаточно просто восстановить значения S-блоков, используемых для шифрования данных. Для этого необходимо найти «нулевой вектор» с использованием нулевого ключа шифрования, что занимает порядка 2^{32} операций шифрования. А затем с использованием этого нулевого вектора вычислить значения S-блоков, что занимает порядка 2^{11} операций. До сих пор остается открытым вопрос, насколько заполнение S-блоков влияет на стойкость алгоритма ГОСТ, хотя считается, что большого количества раундов и избыточной длины ключа достаточно для обеспечения его стойкости.

Для собственно шифрования информации используются режимы гаммирования и гаммирования с обратной связью. В данных режимах шифрование информации производится побитовым сложением по модулю 2 каждого 64-битного блока шифруемой информации с блоком гаммы шифра:

$$Т^{ш} = Т^{о} \oplus Г^{ш}.$$

Гамма шифра – это псевдослучайная последовательность, вырабатываемая с использованием основного преобразования алгоритма ГОСТ 28147-89 следующим образом (рис. 18).

1. В накопители N1 и N2 записывается синхропосылка S длиной 64 бита.

2. Содержимое регистров N1 и N2 шифруется в режиме простой замены.

3. Результат зашифрования из регистров N1 и N2 записывается в регистры N3 и N4 соответственно.

4. Содержимое регистра N4 суммируется по модулю $(2^{32} - 1)$ с содержимым регистра N6, в котором находится константа C1 $(2^{24} + 2^{16} + 2^8 + 2^2)$, а содержимое регистра N3 суммируется по модулю 2^{32} с содержимым регистра N5, в котором находится константа C2 $(2^{24} + 2^{16} + 2^8 + 1)$. Результат сложения записывается соответственно в регистры N4 и N3.

5. Содержимое регистров N3 и N4 записывается в регистры N1 и N2 соответственно, при этом содержимое регистров N3 и N4 сохраняется.

6. Содержимое регистров N1 и N2 шифруется в режиме простой замены, результат записывается в регистры N1 и N2, и их содержимое образует первый 64-битный блок гаммы.

7. Алгоритм генерации остальных блоков гаммы заключается в суммировании содержимого регистров N3 и N4 с содержимым регистров N5 и N6 соответственно, с сохранением результата в N3 и N4, переписыванием содержимого N3 и N4 в N1 и N2 соответственно и последующем шифровании в режиме простой замены содержимого регистров N1 и N2.

Синхропосылка S передается на приемную сторону в открытом или в зашифрованном виде. В некоторых системах связи генерация синхропосылки проходит процедуру согласования между сторонами, участвующими в информационном обмене.

Таким образом, непосредственное шифрование данных происходит в сумматоре СМ5, в то время как вся остальная схема предназначена для выработки гаммы шифра.

Расшифрование происходит аналогично шифрованию путем сложения по модулю 2 блоков выработанной на приемной стороне гаммы и блоков зашифрованного текста:

$$T^{o} = T^{ш} \oplus \Gamma^{ш}.$$

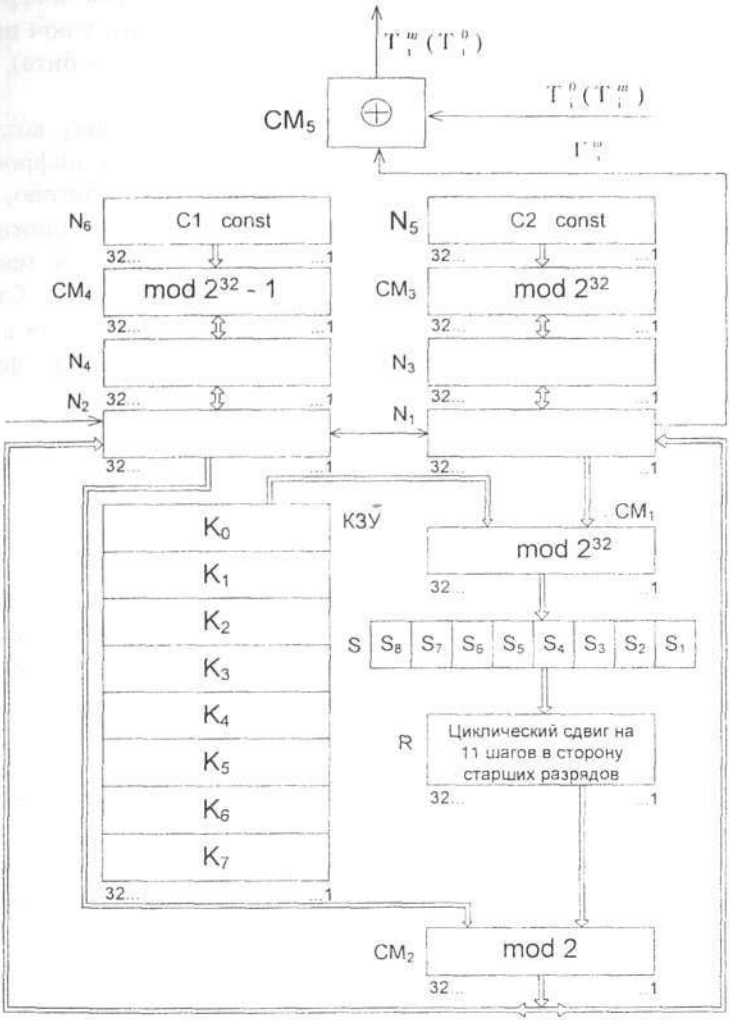


Рис. 18. Режим гаммирования ГОСТ 28147-89

Ясно, что для расшифрования информации необходимо иметь тот же ключ шифрования и то же самое значение синхропосылки, что и при зашифровывании. Существуют реализации алгоритма ГОСТ 28147-89, в которых синхропосылка так-

же является секретным элементом, наряду с ключом шифрования. Фактически в этом случае можно считать, что ключ шифрования увеличивается на длину синхропосылки (64 бита), что усиливает стойкость алгоритма.

Режим гаммирования позволяет злоумышленнику воздействовать на исходный текст путем изменения битов шифрованного текста, так как шифрование производится побитово. Изменив один бит в зашифрованном тексте на противоположный, получим изменение того же бита в расшифрованном тексте. Гаммирование с обратной связью позволяет зацепить блоки один за другой и любое изменение хотя бы одного бита в каком-либо месте шифртекста повлечет за собой при расшифровке повреждение информации во всех последующих блоках, что легко заметить.

Режим гаммирования с обратной связью работает следующим образом (рис. 19).

1. В накопители N1 и N2 записывается синхропосылка S длиной 64 бита.

2. Содержимое регистров N1 и N2 шифруется в режиме простой замены. Полученное в результате зашифрования значение образует первый блок гаммы. Блок гаммы суммируется с блоком открытого текста, тем самым образуя шифр-текст.

3. Полученный шифр-текст записывается в регистры N1 и N2 и происходит возврат к шагу 2.

С помощью режима выработки имитовставок вычисляются имитовставки – криптографические контрольные суммы информации, вычисленные с использованием определенного ключа шифрования. Имитовставки обычно вычисляются до зашифровывания информации и хранятся или отправляются вместе с зашифрованными данными, чтобы впоследствии использоваться для контроля целостности. После расшифровывания информации имитовставка вычисляется снова и сравнивается с хранимой; несовпадение значений указывает на порчу или преднамеренную модификацию данных при хранении или передаче, или на ошибку расшифрования.

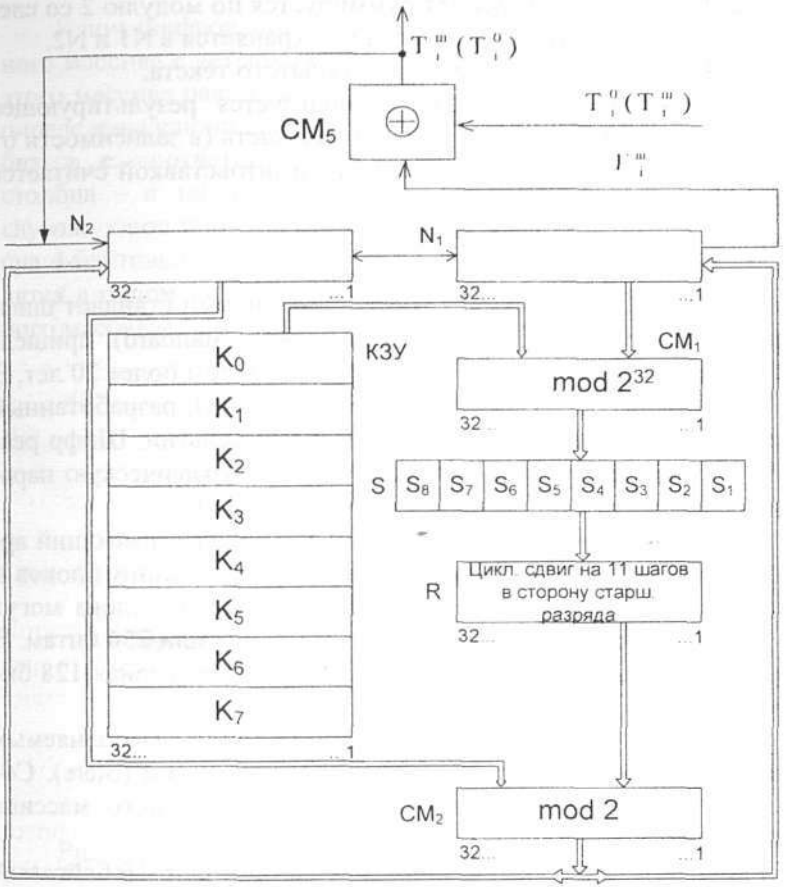


Рис. 19. Режим гаммирования с обратной связью

В режиме выработки имитовставки выполняются следующие операции:

1. Первый 64-битный блок информации, для которой вычисляется имитовставка, записывается в регистры N1 и N2 и зашифровывается в сокращенном режиме простой замены, в котором выполняется 16 раундов основного преобразования вместо 32-х.

2. Полученный результат суммируется по модулю 2 со следующим блоком открытого текста и сохраняется в N1 и N2.

3. И т.д. до последнего блока открытого текста.

В качестве имитовставки используется результирующее содержимое регистров N1 и N2 или его часть (в зависимости от требуемого уровня стойкости). Часто имитовставкой считается 32-битное содержимое регистра N1.

3.3. Стандарт AES

В мае 2002 г. в США вступил в силу новый стандарт шифрования данных AES (Advanced Encryption Standard), пришедший на смену DES, который являлся стандартом более 20 лет. В основе стандарта AES лежит алгоритм Rijndael, разработанный двумя специалистами по криптографии из Бельгии. Шифр реализует совершенно нетрадиционную криптографическую парадигму, полностью отказавшись от сети Фейстеля.

Rijndael – это итерационный блочный шифр, имеющий архитектуру «Квадрат». Шифр имеет переменную длину блоков и различные длины ключей. Длина ключа и длина блока могут быть равны независимо друг от друга 128, 192 или 256 битам. В стандарте AES определена длина блока данных, равная 128 битам.

Промежуточные результаты преобразований, выполняемых в рамках криптоалгоритма, называют состояниями (State). Состояние можно представить в виде прямоугольного массива байтов (рис. 20).

При размере блока, равном 128 битам, этот 16-байтовый массив (рис. 21) имеет 4 строки и 4 столбца (каждая строка и каждый столбец в этом случае могут рассматриваться как 32-разрядные слова). Входные данные для шифра обозначаются как байты состояния в порядке s_{00}, s_{10}, s_{20}, s_{30}, s_{01}, s_{11}, s_{21}, s_{31}, ...

После завершения действия шифра выходные данные получаются из байтов состояния в том же порядке. В общем случае число столбцов N_b равно длине блока, деленной на 32.

На рис. 20 представлен пример 128-разрядного блока данных в виде массива State, где a_i – байт блока данных, а каждый столбец – одно 32-разрядное слово.

Ключ шифрования также представлен в виде прямоугольного массива с четырьмя строками (рис. 22), число столбцов N_k этого массива равно длине ключа, деленной на 32. В стандарте определены ключи всех трех размеров – 128 бит, 192 бита и 256 бит, т. е. соответственно 4, 6 и 8 32-разрядных слова (или столбца – в табличной форме представления). В некоторых случаях ключ шифрования рассматривается как линейный массив 4-байтовых слов. Слова состоят из 4 байтов, которые находятся в одном столбце (при представлении в виде прямоугольного массива).

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

Рис. 20. Пример представления 128-разрядного блока данных

s_{00}	s_{01}	s_{02}	s_{03}
s_{10}	s_{11}	s_{12}	s_{13}
s_{20}	s_{21}	s_{22}	s_{23}
s_{30}	s_{31}	s_{32}	s_{33}

Рис. 21. Формат представления блока входных данных

k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

Рис. 22. Формат данных ключа шифрования

Число раундов N_r в алгоритме Rijndael зависит от значений N_b – длина блока и N_k – длина ключа (табл. 15). В стандарте

AES определено соответствие между размером ключа, размером блока данных и числом раундов шифрования, как показано в табл. 16.

Таблица 15

Число раундов N_r как функция от длины ключа N_k и длины блока N_b

N_r	N_b=4	N_b=6	N_b=8
N_k=4	10	12	14
N_k=6	12	12	14
N_k=8	14	14	14

Таблица 16

Соответствие между длиной ключа, размером блока данных и числом раундов

Стандарт	Длина ключа (N_k слов)	Размер блока данных (N_b слов)	Число раундов (N_r)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Один раунд состоит из четырех различных преобразований:

– замены байтов SubBytes() – побайтовой подстановки в S-блоках с фиксированной таблицей замен размерностью 8x256;

– сдвига строк ShiftRows() – побайтового сдвига строк массива State на различное количество байт;

– перемешивания столбцов MixColumns() – умножение столбцов состояния, рассматриваемых как многочлены над $GF(2^8)$, на многочлен третьей степени g(x) по модулю x^4+1;

– сложения с раундовым ключом AddRoundKey() – поразрядного XOR с текущим фрагментом развернутого ключа.

Преобразование SubBytes() представляет собой нелинейную замену байтов, выполняемую независимо с каждым байтом состояния. Таблицы замены S-блока являются инвертируемыми

и построены из композиции следующих двух преобразований входного байта:

1) получение обратного элемента относительно умножения в поле $GF(2^8)$, нулевой элемент {00} переходит сам в себя;

2) применение преобразования над GF(2), определенного следующим образом:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Суть преобразования может быть описана уравнением $b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i$, где $c_0=c_1=c_5=c_6=1$, $c_2=c_3=c_4=c_7=0$, b_i и b_i' – соответственно исходное и преобразованное значения i-го бита, $i=\overline{0,7}$. Подробный пример преобразования байта данных по вышеописанным формулам можно найти в работах [9, 11].

Применение описанного S-блока ко всем байтам состояния обозначается как SubBytes(State). На рис. 23 иллюстрируется применение преобразования SubBytes() к состоянию State. Логика работы S-блока при преобразовании байта {xy} отражена в табл. 17. Например, результат {fb} преобразования байта {63} находится на пересечении 7-й строки и 4-го столбца (выделено в таблице серым цветом).

Преобразование сдвига строк (ShiftRows) выглядит следующим образом: последние 3 строки состояния циклически сдвигаются влево на различное число байт. Строка 1 сдвигается на C_1 байт, строка 2 – на C_2 байт, и строка 3 – на C_3 байт. Значе-

ния сдвигов C_1, C_2 и C_3 в Rijndael зависят от длины блока N_b. Их величины приведены в табл. 18.

Таблица 17

Таблица замен S-блока

X	Y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

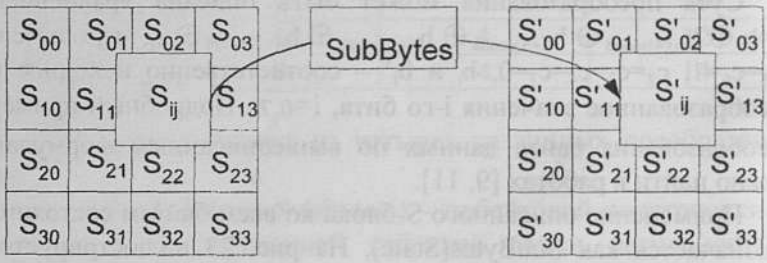


Рис. 23. Применение преобразования SubBytes()

Таблица 18

Величина сдвига для блоков разной длины

N_b	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

В стандарте AES, где определен единственный размер блока, равный 128 битам, $C_1 = 1$, $C_2 = 2$, $C_3 = 3$.

Операция сдвига последних трех строк состояния обозначена как ShiftRows(State). На рис. 24 показано влияние преобразования на состояние.

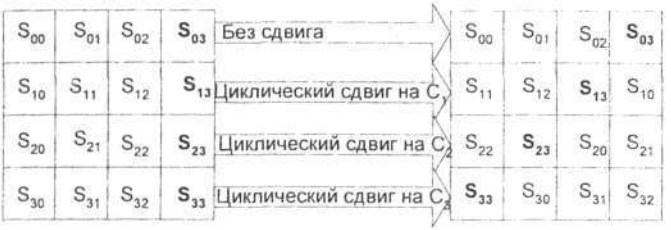


Рис. 24. Действие операции ShiftRows() на строки состояния

Преобразование перемешивания столбцов (MixColumns) – это такое преобразование, при котором столбцы состояния рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю x^4+1 на многочлен g(x), выглядящий следующим образом: $g(x)=\{03\}x^3+\{01\}x^2+\{01\}x+\{02\}$.

Это может быть представлено в матричном виде так:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, 0 \le c \le 3,$$

где c – номер столбца массива State.

В результате такого умножения байты столбца s_{0c}, s_{1c}, s_{2c}, s_{3c} заменяются соответственно на байты

$$s'_{0c}=(\{02\}*s_{0c})\oplus(\{03\}*s_{1c})\oplus s_{2c}\oplus s_{3c},$$
$$s'_{1c}=s_{0c}\oplus(\{02\}*s_{1c})\oplus(\{03\}*s_{2c})\oplus s_{c3},$$
$$s'_{2c}=s_{0c}\oplus s_{1c}\oplus(\{02\}*s_{2c})\oplus(\{03\}*s_{3c}),$$
$$s'_{3c}=(\{03\}*s_{0c})\oplus s_{1c}\oplus s_{2c}\oplus(\{02\}*s_{3c}).$$

Применение этой операции ко всем четырем столбцам состояния обозначено как MixColumns().Подробный пример преобразования столбца данных с помощью операции MixColumns() можно найти в работах [9, 11]. На рис. 25 демонстрируется применение преобразования MixColumns() к столбцу состояния.

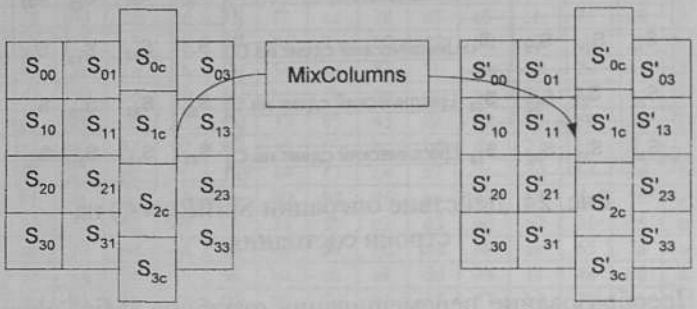


Рис. 25. Действие операции MixColumns() на столбцы состояния

В операции добавления раундового ключа (AddRoundKey) раундовый ключ добавляется к состоянию посредством простого поразрядного XOR.

Раундовый ключ вырабатывается из ключа шифрования посредством алгоритма выработки ключей (keyshedule). Длина раундового ключа (в 32-разрядных словах) равна длине блока N_b.

Преобразование, содержащее добавление посредством операции сложения по модулю 2 (XOR) раундового ключа к состоянию, показано на рис. 26, обозначено как AddRoundKey(State, RoundKey).

Раундовые ключи получаются из ключа шифрования посредством алгоритма выработки ключей. Он содержит два компонента: расширение ключа (Key Expansion) и выбор раундового ключа (Round Key Selection). Основополагающие принципы алгоритма выглядят следующим образом:

– общее число битов раундовых ключей равно длине блока, умноженной на число раундов, плюс 1 (например, для дли-

ны блока 128 бит и 10 раундов требуется 1408 бит раундовых ключей);

– ключ шифрования расширяется в расширенный ключ (Expanded Key);

– раундовые ключи берутся из расширенного ключа следующим образом: первый раундовый ключ содержит первые N_b слов, второй – следующие N_b слов и т. д.;

– расширенный ключ (Key Expansion) в Rijndael представляет собой линейный массив w[i] из $N_b(N_r+1)$ 4-байтовых слов, $i=\overline{0,4(N_r+1)}$;

– первые N_k слов содержат ключ шифрования. Все остальные слова определяются рекурсивно из слов с меньшими индексами. Алгоритм выработки подключей зависит от величины N_k;

– первые N_k слов заполняются ключом шифрования (рис. 27). Каждое последующее слово w[i] получается посредством сложения по модулю 2 предыдущего слова w[i–1] и слова на N_k позиций ранее, т. е. w[i–N_k]:

$$w[i] = w[i-1] \oplus w[i-N_k].$$

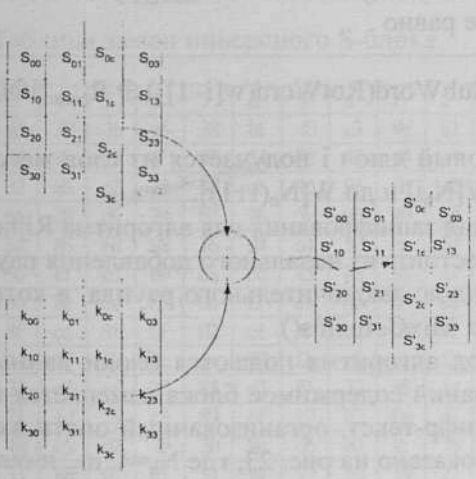


Рис. 26. Сложение данных с раундовым подключом

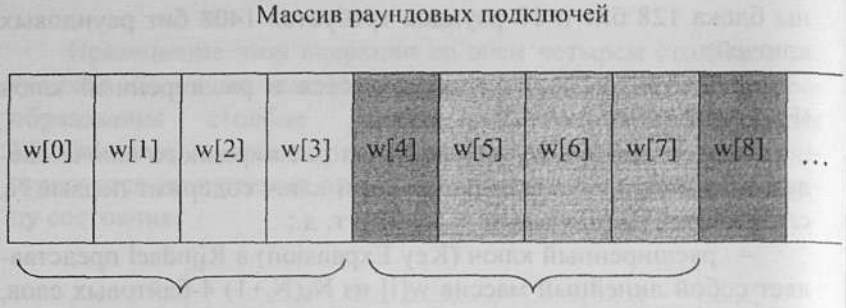


Рис. 27. Выработка раундовых подключей

Для слов, позиция которых кратна N_k, перед операцией сложения по модулю 2 применяется преобразование к w[i–1], а затем еще прибавляется раундовая константа R_{const}. Преобразование реализуется с помощью двух дополнительных функций: RotWord(), осуществляющей побайтовый сдвиг 32-разрядного слова по формуле $\{a_0, a_1, a_2, a_3\} \rightarrow \{a_1, a_2, a_3, a_0\}$, и SubWord(), осуществляющей побайтовую замену с использованием S-блока функции SubBytes(). Значение $R_{const}[j]$ равно 2^{j-1}. Значение w[i] в этом случае равно

$$w[i] = \text{SubWord}(\text{RotWord}(w[i-1])) \oplus R_{const}[i/N_k] \oplus w[i-N_k].$$

Раундовый ключ i получается из слов массива раундового ключа от $W[N_b i]$ и до $W[N_b(i+1)]$.

Функция зашифрования для алгоритма Rijndael показана на рис. 28 и состоит: из начального добавления раундового ключа; (Nr–1) раундов; заключительного раунда, в котором отсутствует операция MixColumns().

На вход алгоритма подаются блоки данных State, в ходе преобразований содержимое блока изменяется и на выходе образуется шифр-текст, организованный опять же в виде блоков State, как показано на рис. 23, где $N_b=4$, in_m и out_m – байты соответственно входного и выходного блоков, $m=\overline{0,15}$, s_{ij} – байт,

находящийся на пересечении i-й строки и j-го столбца массива State, $i=\overline{0,3}$, $j=\overline{0,3}$.

Перед началом первого раунда происходит суммирование по модулю 2 с начальным ключом шифрования, затем – преобразование массива байтов State в течение 10, 12 или 14 раундов в зависимости от длины ключа. Последний раунд несколько отличается от предыдущих тем, что не задействует функцию перемешивания байтов в столбцах MixColumns().

Если вместо SubBytes(), ShiftRows(), MixColumns() и AddRoundKey() в обратной последовательности выполнить инверсные им преобразования, можно построить функцию обратного расшифрования. При этом порядок использования раундовых ключей является обратным по отношению к тому, который используется при зашифровании.

Функция AddRoundKey() обратна сама себе, учитывая свойства используемой в ней операции сложения по модулю 2 (XOR).

Логика работы инверсного S-блока InvSubBytes при преобразовании байта {xy} отражена в табл. 19.

Таблица 19

Таблица замен инверсного S-блока

x	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

В преобразовании InvShiftRows последние 3 строки состояния сдвигаются вправо на различное число байтов. Строка 1 сдвигается на C_1 байт, строка 2 – на C_2 байт, и строка 3 – на C_3 байт. Значение сдвигов C_1, C_2 и C_3 зависят от длины блока N_b.

В преобразовании InvMixColumns столбцы состояния рассматриваются как многочлен над $GF(2^8)$ и умножаются по модулю x^4+1 на многочлен $g^{-1}(x)$, выглядящий следующим образом:

$$g^{-1}(x)=\{0b\}x^3+\{0d\}x^2+\{09\}x+\{0e\}.$$

Это может быть представлено в матричном виде так:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 9d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, \quad 0 \le c \le 3.$$

В результате на выходе получаются следующие байты:

$$s'_{0c}=(\{0e\}*s_{0c})\oplus(\{0b\}*s_{1c})\oplus(\{0d\}*s_{2c})\oplus(\{09\}*s_{3c}),$$
$$s'_{1c}=(\{09\}*s_{0c})\oplus(\{0e\}*s_{1c})\oplus(\{0b\}*s_{2c})\oplus(\{0d\}*s_{3c}),$$
$$s'_{2c}=(\{0d\}*s_{0c})\oplus(\{09\}*s_{1c})\oplus(\{0e\}*s_{2c})\oplus(\{0b\}*s_{3c}),$$
$$s'_{3c}=(\{0b\}*s_{0c})\oplus(\{0d\}*s_{1c})\oplus(\{09\}*s_{2c})\oplus(\{0e\}*s_{3c}).$$

Алгоритм обратного расшифрования, описанный выше, имеет порядок приложений операций-функций, обратный порядку операций в алгоритме прямого расшифрования, но использует те же параметры (развернутый ключ). Однако некоторые свойства алгоритма шифрования Rijndael позволяют применить для расшифрования тот же порядок приложения функций (обратных, используемых для зашифрования) за счет изменения некоторых параметров, а именно – развернутого ключа.

Два следующих свойства позволяют применить алгоритм прямого расшифрования.

Порядок приложений функций SubBytes() и ShiftRows() не играет роли. То же самое верно и для операций InvSubBytes() и InvShiftRows(). Это происходит потому, что функции SubBytes() и InvSubBytes() работают с байтами, а операции ShiftRows() и InvShiftRows() сдвигают целые байты, не затрагивая их значения.

Операция MixColumns() является линейной относительно входных данных, что означает: InvMixColumns(State XOR XOR RoundKey) = InvMixColumns(State) XOR InvMixColumns(RoundKey).

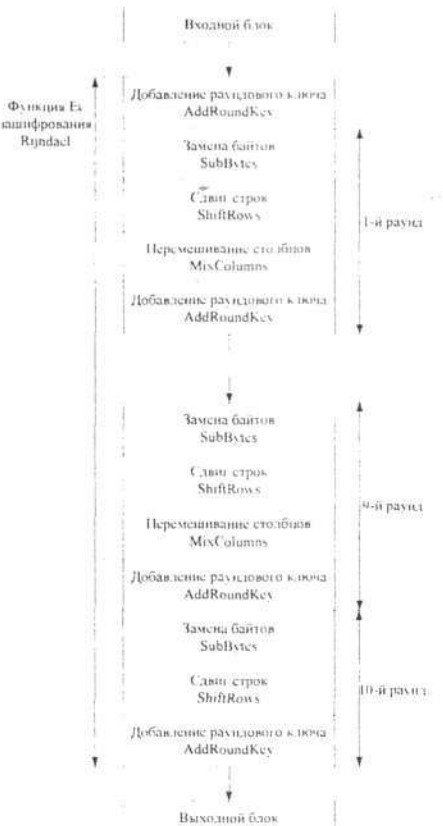


Рис. 28. Алгоритм шифрования Rijndael

Эти свойства функций алгоритма шифрования позволяют изменить порядок применения функций InvSubBytes() и InvShiftRows(). Функции AddRoundKey() и InvMixColumns() также могут быть применены в обратном порядке, но при условии, что столбцы (32-битные слова) развернутого ключа расшифрования предварительно пропущены через функцию InvMixColumns().

В табл. 20 приведена процедура зашифрования, а также два эквивалентных варианта процедуры расшифрования при использовании двухраундового варианта Rijndael. Первый вариант функции расшифрования суть обычная инверсия функции зашифрования. Второй вариант функции зашифрования получен из первого после изменения порядка следования операций в трех парах преобразований:

InvShiftRows – InvSubBytes(дважды) и AddRoundKey – InvMixColumns.

Таблица 20

Последовательность преобразований в двухраундовом варианте Rijndael

Функция зашифрования двухраундового варианта Rijndael	Функция обратного расшифрования двухраундового варианта Rijndael	Эквивалентная функция прямого расшифрования двухраундового варианта Rijndael
AddRoundKey	AddRoundKey	AddRoundKey
SubBytes	InvShiftRows	InvSubBytes
ShiftRows	InvSubBytes	InvShiftRows
MixColumns	AddRoundKey	InvMixColumns
AddRoundKey	InvMixColumns	AddRoundKey
SubBytes	InvShiftRows	InvSubBytes
ShiftRows	InvSubBytes	InvShiftRows
AddRoundKey	AddRoundKey	AddRoundKey

Очевидно, что результат преобразования при переходе от исходной к обратной последовательности выполнения операций в указанных парах не изменится.

Видно, что процедура зашифрования и второй вариант процедуры расшифрования совпадают с точностью до порядка использования раундовых ключей (при выполнении операции AddRoundKey), таблиц замен (при выполнении операции SubBytes) и матриц преобразования (при выполнении операций MixColumns и InvMixColumns). Данный результат легко обобщить и на любое другое число раундов.

В основе алгоритма лежит новая архитектура «Квадрат», обеспечивающая быстрое рассеивание и перемешивание информации, при этом за один раунд преобразованию подвергается весь входной блок. За счет того, что в процедуре SubBytes() блоки данных могут быть преобразованы независимо друг от друга, а в процедуре MixColumns() независимо ведется обработка столбцов состояния, алгоритм Rijndael может быть легко распараллелен.

3.4. Проект нового стандарта шифрования данных ГОСТ – алгоритм «Кузнечик»

В настоящее время идет активное обсуждение разработки и принятия нового отечественного стандарта шифрования данных. Проект стандарта с подробным описанием представлен на сайте http://www.tc26.ru/

Проект стандарта включает описание двух разных блочных шифров. Первый – это блочный шифр, описанный в ГОСТ 28147-89 с зафиксированными блоками нелинейной подстановки. Второй – новый блочный шифр типа «подстановочно-перестановочная сеть» с размером блока 128 бит, для генерации раундовых ключей которого используется сеть Фейстеля. Данный тип шифров является хорошо изученным и относительно простым с точки зрения криптографического анализа и обоснования требуемых свойств. Новый блочный шифр в проекте стандарта описан под именем «Кузнечик» («Kuznyechik»).

Работа алгоритма «Кузнечик» происходит следующим образом. На вход шифрования поступает 128-битовый блок от-

крытого текста. Первоначально данный текст складывается по модулю 2 с первым раундовым подключом. После этого выполняется девять раундов преобразования. Каждый раунд состоит из трех операций: преобразования S, преобразования L, сложения с очередным раундовым подключом. Выход девятого раунда образует блок шифр-текста длиной 128 бит. Алгоритм расшифрования использует инверсные операции. Кроме того, меняется порядок использования раундовых подключей. На вход алгоритма расшифрования «Кузнечик» поступает 128-битовый блок шифрованного текста. Первоначально данный текст складывается по модулю 2 с последним раундовым подключом K10. После этого выполняется девять раундов преобразования. Каждый раунд состоит из трех операций: инверсного преобразования S, инверсного преобразования L, сложения с очередным раундовым подключом. Выход девятого раунда образует блок восстановленного открытого текста длиной 128 бит. Общий вид алгоритма шифрования «Кузнечик» можно представить так, как это сделано на рис. 29.

В алгоритме «Кузнечик» используются следующие математические преобразования. Нелинейное биективное преобразование представляет собой табличную замену. Каждое входное значение определяет индекс массива π, и элемент, стоящий в данной позиции, образует выходное значение.

Линейное преобразование предназначено для перемешивания информации и представляет собой перемножение многочленов, определяемых входом преобразования, с фиксированными константами. Операция умножения осуществляется в поле F, определенном для данного шифра.

Преобразование S является аналогом S-блока замены. На вход преобразования S поступает 128-битное значение, которое разбивается на шестнадцать элементов по 8 бит $a_{15}, \ldots, a_0$. Каждый элемент заменяется с помощью нелинейного биективного преобразования π. Второе базовое преобразование нового шифра – операция R. На выходе преобразования R первый элемент получается с помощью линейного преобразования ℓ, а остальные элементы переписываются в неизменном виде с a_{15} по a_1. Смысл операции L заключается в том, что операция R применя-

ется 16 раз подряд. Таким образом, после шестнадцатикратного применения преобразования R, все байты преобразуемой последовательности изменят свои значения в результате преобразования ℓ.

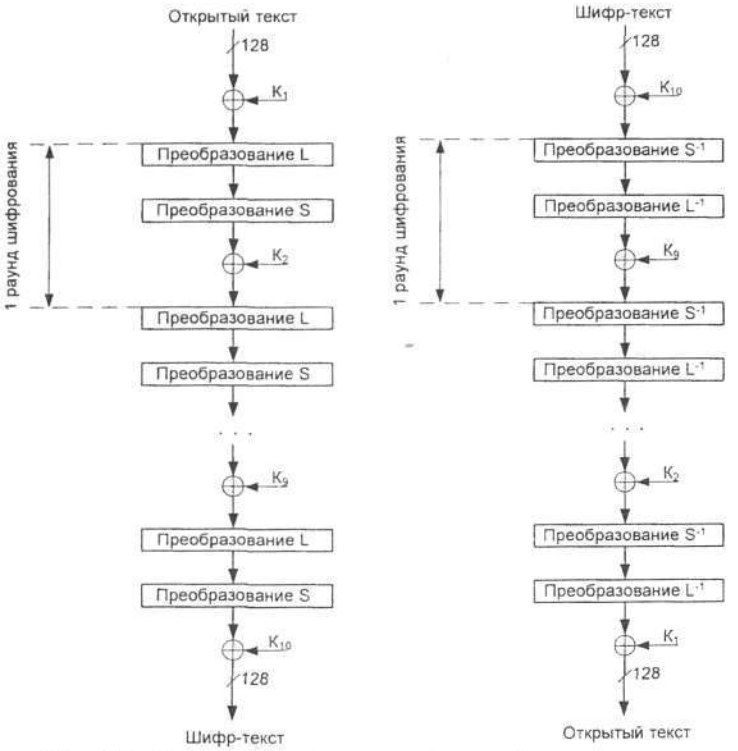


Рис. 29. Схема зашифрования/расшифрования данных с помощью шифра «Кузнечик»

При выработке подключей используется исходный секретный ключ. Его значение образует первые два раундовых подключа. Для вычисления каждой следующей пары подключей используется классическая схема Фейстеля так, как это показано на рис. 30.

Обработка выполняется в течение 8 раундов. Функция F заключается в сложении с константой C и выполнении преобра-

зований S и L. Всего для работы алгоритма шифрования требуется 10 раундовых подключей, поэтому схема, приведенная на рис. 30, применяется 4 раза, вырабатывая каждый раз по 2 раундовых подключа.

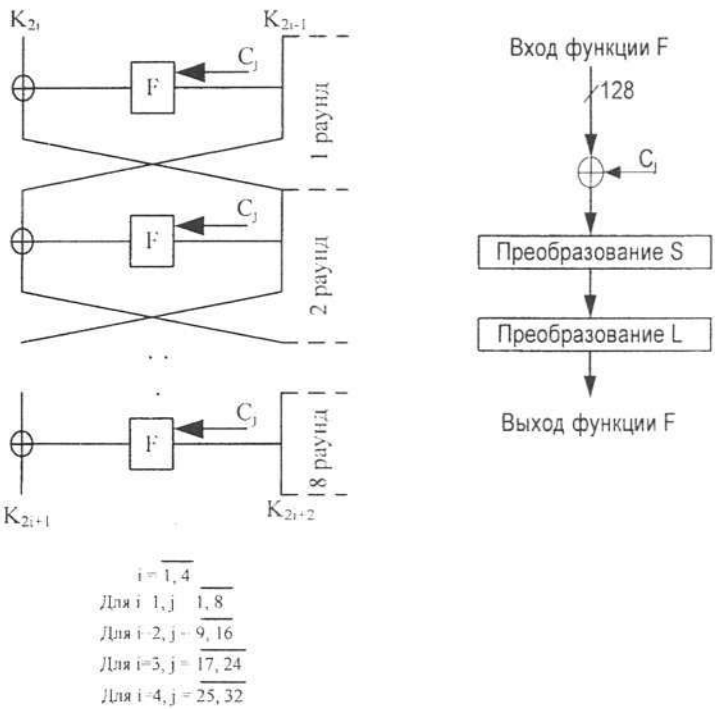


Рис. 30. Схема выработки раундовых подключей

Рассмотрим, как работают основные операции, входящие в состав шифра «Кузнечик». При этом двоичные строки из V_*, длина которых кратна 4, будем записывать в шестнадцатеричном виде, а символ конкатенации ("||") опускать. То есть, строка $a \in V_{4n}$ будет представлена в виде $a_{n-1}a_{n-2}\ldots a_0$, где $a_i \in \{0, 1, \ldots, 9, a, b, c, d, e, f\}$, $i = 0, 1, \ldots, n-1$. Соответствие

между двоичными строками длиной 4 и шестнадцатеричными строками длиной 1 задаётся естественным образом. Преобразование, ставящее в соответствие двоичной строке длиной 4n шестнадцатеричную строку длиной n, и соответствующее обратное преобразование для простоты записи опускаются.

В алгоритме используется операция $S: V_{128} \rightarrow V_{128}$, работа которой описывается следующим образом:

$$S(a) = S(a_{15}||\ldots||a_0) = \pi(a_{15})||\ldots||\pi(a_0),$$

где $a = a_{15}||\ldots||a_0 \in V_{128}$, $a_i \in V_8$, $i = 0, 1, \ldots, 15$.

По сути дела, данное преобразование является аналогом S-блока замены. На вход преобразования поступает 128-битное значение, которое разбивается на шестнадцать элементов по 8 бит $a_{15}, \ldots, a_0$. Каждое значение a_i определяет индекс массива π:

π = (252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77, 233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241, 187, 20, 205, 95, 193, 249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152, 127, 212, 211, 31, 235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162, 253, 58, 206, 204, 181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178, 177, 50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125, 105, 213, 149, 59, 7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228, 136, 217, 231, 137, 225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133, 97, 32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182).

Элемент, стоящий в позиции, определяемой индексом, будет являться выходом преобразования. Схематично процесс

преобразования данных с помощью преобразования S можно сделать так, как это показано на рис. 31.

Так, например, в результате применения преобразования S, значение ffeeddccbbaa99881122334455667700 будет преобразовано в значение b66cd8887d38e8d77765aeea0c9a7efc. Разберем подробнее, как выполнено данное преобразование. Легко можно видеть, что первые два символа ff (что в 10-ричном виде соответствует значению 255) будут указывать на последний элемент массива π. π(255) = 182, что в 16-ричном виде соответствует значению b6, которое и составляет первые символы выходного преобразования. Аналогичным образом видно, что последние два символа 00 будут указывать на первый элемент массива π. π(0) = 252, что в 16-ричном виде соответствует значению fc, которое и составляет последние символы выходного преобразования.

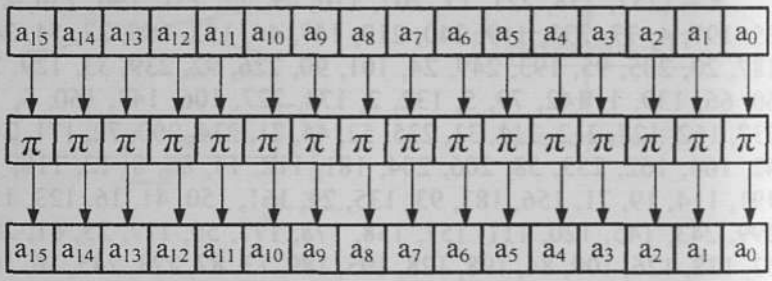


Рис. 31. Преобразование S

Операция R работает следующим образом:

$$R: V_{128} \rightarrow V_{128},$$
$$R(a) = R(a_{15}||\ldots||a_0) = \ell(a_{15}, \ldots, a_0)||a_{15}||\ldots||a_1,$$

где $a = a_{15}||\ldots||a_0 \in V_{128}$, $a_i \in V_8$, $i = 0, 1, \ldots, 15$.

Таким образом, на выходе преобразования R первый элемент будет получен с помощью преобразования ℓ, а остальные элементы перепишутся в неизменном виде с a_{15} по a_1. Схематично процесс преобразования данных с помощью преобразо-

вания R можно сделать так, как это показано на рис. 31. При этом сама операция ℓ работает следующим образом:

$$\ell\,(a_{15}, \ldots, a_0) = \nabla(148 \cdot \Delta(a_{15}) + 32 \cdot \Delta(a_{14}) + 133 \cdot \Delta(a_{13}) + 16 \cdot \Delta(a_{12}) + 194 \cdot \Delta(a_{11}) + 192 \cdot \Delta(a_{10}) + 1 \cdot \Delta(a_9) + 251 \cdot \Delta(a_8) + 1 \cdot \Delta(a_7) + 192 \cdot \Delta(a_6) + 194 \cdot \Delta(a_5) + 16 \cdot \Delta(a_4) + 133 \cdot \Delta(a_3) + 32 \cdot \Delta(a_2) + 148 \cdot \Delta(a_1) + 1 \cdot \Delta(a_0)),$$

для любых $a_i \in V_8$, $i = 0, 1, \ldots, 15$, где операции сложения и умножения осуществляются в поле $\mathbb{F}$, константы являются элементами поля, а Δ обозначает биективное отображение.

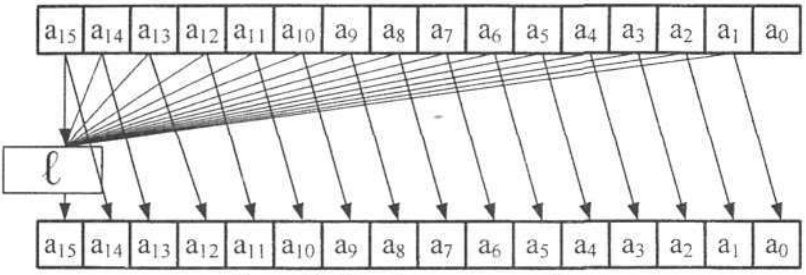


Рис. 32. Преобразование R

Прежде чем перейти к рассмотрению примера, заметим, что все операции для алгоритма «Кузнечик» выполняются в конечном поле $GF(2)[x]/p(x)$, где $p(x) = x^8 + x^7 + x^6 + x + 1$, $p(x) \in GF(2)[x]$; элементы поля F представляются целыми числами, причем элементу $z_0 + z_1 \cdot \theta + \ldots + z_7 \cdot \theta^7 \in F$, где $z_i \in \{0, 1\}$, $i = 0, 1, \ldots, 7$, и θ обозначает класс вычетов по модулю $p(x)$, содержащий x, соответствует число $z_0 + 2 \cdot z_1 + \ldots + 2^7 \cdot z_7$. Это значит, что если в результате применения операций умножения, результат выйдет за пределы степени поля, то для его нормализации необходимо будет вычислить его значение модуля, применив многочлен $p(x) = x^8 + x^7 + x^6 + x + 1$.

В преобразовании $\ell\,(a_{15}, \ldots, a_0)$ для умножения используются константы, приведенные в десятичном виде. Однако, так как операции умножения производятся в поле $GF(2)[x]/p(x)$, где $p(x) = x^8 + x^7 + x^6 + x + 1 \in GF(2)[x]$, то для корректного выпол-

нения умножения необходимо использовать полиномы, соответствующие биективному отображению, сопоставляющие двоичной строке из V_8 элемент поля F. Поэтому, для удобства дальнейшей работы, составим табл. 21, в которой сопоставим десятичные значения констант, их шестнадцатеричный и двоичный вид, а также соответствующие им полиномы.

Таблица 21

Соответствие констант и полиномов для преобразования $\ell(a_{15},\ldots,a_0)$

10-ный вид	16-ный вид	2-ный вид	Полином
1	1	00000001	1
16	f	00001111	$x^3 + x^2 + x + 1$
32	20	00100000	x^5
133	85	10000101	$x^7 + x^2 + 1$
148	94	10010100	$x^7 + x^4 + x^2$
192	c0	11000000	$x^7 + x^6$
194	c2	11000010	$x^7 + x^6 + x$
251	fb	11111011	$x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$

Рассмотрим, как будет работать преобразование R, если значение 00000000000000000000000000000100 поступит на его вход. Значение, поступающее, на вход преобразования R можно представить в виде следующих 16 значений: $a_{15} = 0$, $a_{14} = 0$, $a_{13} = 0$, $a_{12} = 0$, $a_{11} = 0$, $a_{10} = 0$, $a_9 = 0$, $a_8 = 0$, $a_7 = 0$, $a_6 = 0$, $a_5 = 0$, $a_4 = 0$, $a_3 = 0$, $a_2 = 0$, $a_1 = 1$, $a_0 = 0$. Тогда:

$$\ell(a_{15}, \ldots, a_0) = \nabla(148 \cdot 0 + 32 \cdot 0 + 133 \cdot 0 + 16 \cdot 0 + 194 \cdot 0 + +192 \cdot 0 + 1 \cdot 0 + 251 \cdot 0 + 1 \cdot 0 + 192 \cdot 0 + 194 \cdot 0 + 16 \cdot 0 + 133 \cdot 0 + + 32 \cdot 0 + 148 \cdot \Delta(1) + 1 \cdot 0) = \nabla(148 \cdot \Delta(1)).$$

Как упоминалось ранее, Δ обозначает биективное отображение, сопоставляющее двоичной строке из V_8 элемент поля F следующим образом: строке $z_7||\ldots||z_1||z_0$, $z_i \in \{0, 1\}$, $i = 0, 1, \ldots, 7$, соответствует элемент $z_0 + z_1 \cdot \theta + \ldots + z_7 \cdot \theta^7 \in F$. То есть значе-

нию 1 будет соответствовать 1 (см. табл. 21). В таком случае имеем:

$$\nabla(148 \cdot \Delta(1)) = \nabla((x^7 + x^4 + x^2) \cdot 1) = 100101000 = 94_{16}.$$

Значение 148 в десятичном виде соответствует 16-ричному значению 94, которое и является первым элементом выходной последовательности в рассматриваемом примере. Остальные выходные элементы преобразования R просто записаны в виде последовательности значений $a_{15}, \ldots, a_1$:

$$R(00000000000000000000000000000100) =$$
$$=94000000000000000000000000000001.$$

В данном случае умножение выполнялось на 1 и в результате умножения было образовано всего одно слагаемое. Рассмотрим пример посложнее. Применим к полученному результату еще раз преобразование R.

В этом случае значение, поступающее на вход преобразования R, можно представить в виде следующих 16 значений: $a_{15} = 94$, $a_{14} = 0$, $a_{13} = 0$, $a_{12} = 0$, $a_{11} = 0$, $a_{10} = 0$, $a_9 = 0$, $a_8 = 0$, $a_7 = 0$, $a_6 = 0$, $a_5 = 0$, $a_4 = 0$, $a_3 = 0$, $a_2 = 0$, $a_1 = 0$, $a_0 = 1$. Тогда запишем:

$$\ell(a_{15}, \ldots, a_0) = \nabla(148 \cdot \Delta(94) + 32 \cdot 0 + 133 \cdot 0 + 16 \cdot 0 + 194 \cdot 0 +$$
$$+192 \cdot 0 + 1 \cdot 0 + 251 \cdot 0 + 1 \cdot 0 + 192 \cdot 0 + 194 \cdot 0 + 16 \cdot 0 + 133 \cdot 0 +$$
$$+32 \cdot 0 + 148 \cdot 0 + 1 \cdot 1) = \nabla(148 \cdot 148 + 1) = \nabla(148 \cdot \Delta(94) + 1 \cdot 1).$$

Значение 148 в десятичном виде соответствует 16-ричному значению 94, поэтому полиномы для двух этих значений будут одинаковые. И для получения результата их надо перемножить между собой. При этом необходимо отметить, что при образовании полиномов операция сложения понимается как операция сложения по модулю 2. Таким образом, члены полинома, полученного в результате умножения, имеющие одинаковую степень, будут взаимоисключаться:

$$148 \cdot \Delta(94) = (x^7 + x^4 + x^2) \cdot (x^7 + x^4 + x^2) = x^{14} + \underline{x^{11}} + \underline{x^9} + \underline{x^{11}} + x^8 + \underline{x^6} + \underline{x^9} + \underline{x^6} + x^4 = x^{14} + x^8 + x^4.$$

Степень полученного полинома превышает допустимую степень полинома в поле GF(2)[x]/p(x). Поэтому полученное произведение необходимо разделить на значение полинома p(x) и в качестве результата взять остаток от деления так, как показано на рис. 33.

Тогда получаем, что

$$148 \cdot \Delta(94) = x^{14} + x^8 + x^4 = x^7 + x^5 + x^2.$$

Следовательно:

$$\ell(a_{15}, \ldots, a_0) = \nabla(148 \cdot \Delta(94) + 1 \cdot 1) = \nabla(x^7 + x^5 + x^2 + 1) = 10100101 = a5.$$

Таким образом получается, что

$$R(94000000000000000000000000000001) = a5940000000000000000000000000000.$$

Ниже приведено еще два примера преобразования, которые легко можно вычислить, воспользовавшись описанными выше приемами:

$$R(a5940000000000000000000000000000) = 64a5940000000000000000000000000000,$$
$$R(64a5940000000000000000000000000000) = 0d64a5940000000000000000000000000000.$$

Операция L опирается на преобразование R и выглядит следующим образом:

$$L: V_{128} \rightarrow V_{128};$$
$$L(a) = R^{16}(a), \text{ где } a \in V_{128}.$$

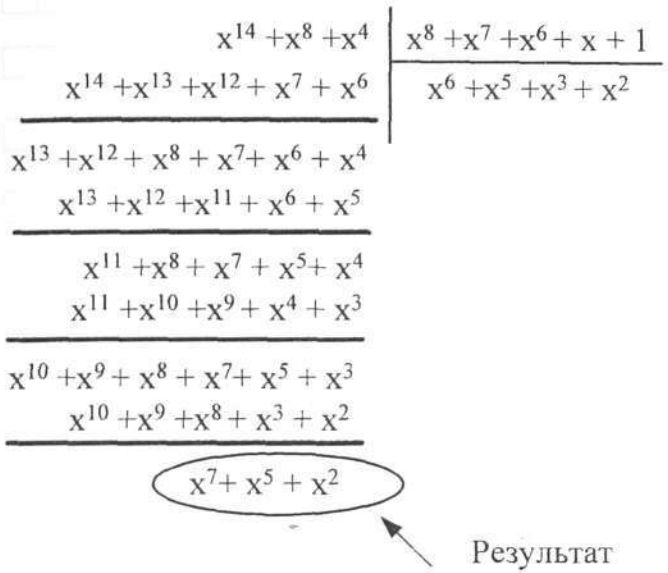


Рис. 33. Нормализация полинома

По сути дела смысл операции L заключается в том, что операция R применяется 16 раз подряд. Если изобразить операцию L схематично так, как показано на рис. 33, то станет ясно, что после шестнадцатикратного применения преобразования R все байты преобразуемой последовательности изменят свои значения в результате преобразования ℓ. На рис. 34 упрощены связи между состояниями R^i и не отражены связи так, как это показано на рис. 32 (однако они подразумеваются). На рис. 34 черными квадратиками выделены байты, которые изменяют свои значения в результате очередного применения преобразования ℓ.

Так, чтобы выполнить преобразование

$$L(64a594000000000000000000000000) = \\ = d456584dd0e3e84cc3166e4b7fa2890d,$$

необходимо выполнить 16 промежуточных итераций следующим образом:

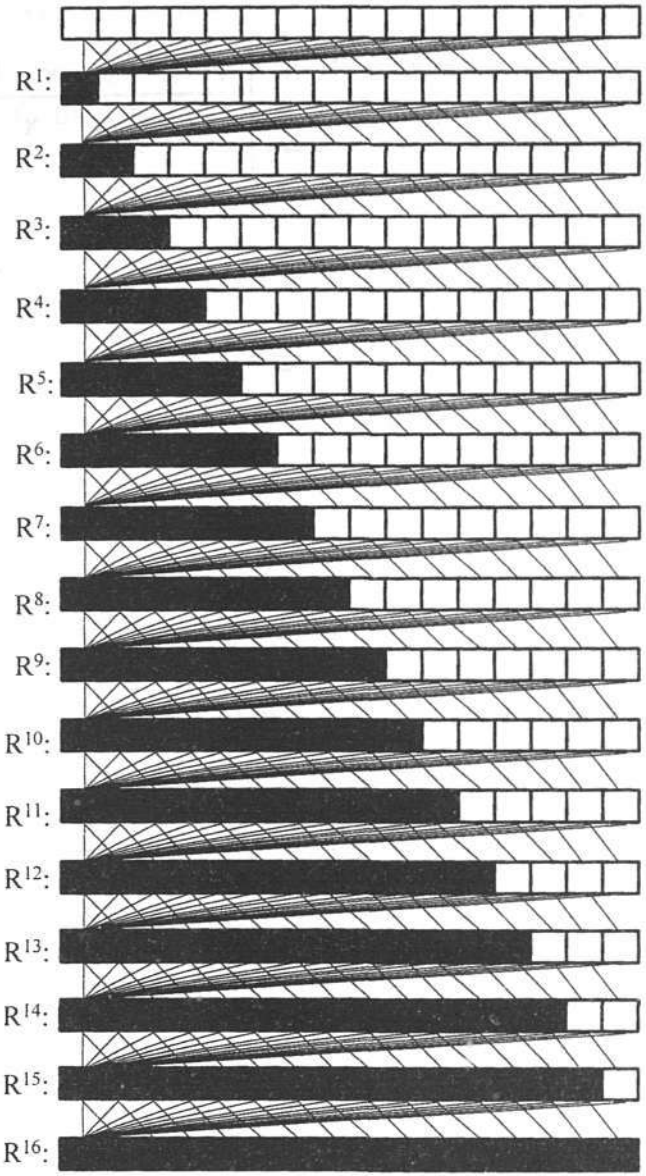


Рис. 34. Преобразование L

L(64a594000000000000000000000000)
R1: 0d 64 a5 94 00 00 00 00 00 00 00 00 00 00 00 00
R2: 89 0d 64 a5 94 00 00 00 00 00 00 00 00 00 00 00
R3: a2 89 0d 64 a5 94 00 00 00 00 00 00 00 00 00 00
R4: 7f a2 89 0d 64 a5 94 00 00 00 00 00 00 00 00 00
R5: 4b 7f a2 89 0d 64 a5 94 00 00 00 00 00 00 00 00
R6: 6e 4b 7f a2 89 0d 64 a5 94 00 00 00 00 00 00 00
R7: 16 6e 4b 7f a2 89 0d 64 a5 94 00 00 00 00 00 00
R8: c3 16 6e 4b 7f a2 89 0d 64 a5 94 00 00 00 00 00
R9: 4c c3 16 6e 4b 7f a2 89 0d 64 a5 94 00 00 00 00
R10: e8 4c c3 16 6e 4b 7f a2 89 0d 64 a5 94 00 00 00
R11: e3 e8 4c c3 16 6e 4b 7f a2 89 0d 64 a5 94 00 00
R12: d0 e3 e8 4c c3 16 6e 4b 7f a2 89 0d 64 a5 94 00
R13: 4d d0 e3 e8 4c c3 16 6e 4b 7f a2 89 0d 64 a5 94
R14: 58 4d d0 e3 e8 4c c3 16 6e 4b 7f a2 89 0d 64 a5
R15: 56 58 4d d0 e3 e8 4c c3 16 6e 4b 7f a2 89 0d 64
R16: d4 56 58 4d d0 e3 e8 4c c3 16 6e 4b 7f a2 89 0d

Ниже приведено несколько примеров выполнения преобразования L:

$$L(d456584dd0e3e84cc3166e4b7fa2890d) = 79d26221b87b584cd42fbc4ffea5de9a,$$
$$L(79d26221b87b584cd42fbc4ffea5de9a) = 0e93691a0cfc60408b7b68f66b513c13,$$
$$L(0e93691a0cfc60408b7b68f66b513c13) = e6a8094fee0aa204fd97bcb0b44b8580.$$

Параллельно с проектом стандарта обсуждается проект о принятии стандарта на использование различных режимов шифрования для блочных шифров, и в первую очередь для нового стандарта ГОСТ. В данном проекте стандарта определены следующие режимы работы алгоритмов блочного шифрования:

– режим простой замены (ElectronicCodebook, ECB);
– режим гаммирования (Counter, CTR);
– режим гаммирования с обратной связью по выходу (OutputFeedback, OFB);

– режим простой замены с зацеплением (CipherBlock-Chaining, CBC);

– режим гаммирования с обратной связью по шифртексту (CipherFeedback, CFB);

– режим выработки имитовставки (Message Authentication Code algorithm).

Данные режимы могут использоваться в качестве режимов для блочных шифров с произвольной длиной блока n.

Режимы блочного шифра выведены в отдельный проект национального стандарта «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров», что соответствует международной практике. Ожидается, что описанный блочный шифр с длиной блока 128 бит будет устойчив ко всем известным на сегодняшний день атакам на блочные шифры. Фиксация блоков нелинейной подстановки сделает алгоритм, описанный в стандарте ГОСТ 28147-89, более унифицированным и поможет исключить использование «слабых» блоков нелинейной подстановки [12].

4. УЧЕБНЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ

4.1. Учебный алгоритм шифрования УАШ

4.1.1. Предпосылки создания

Затруднительно использовать в учебных целях один из известных на сегодняшний день алгоритмов блочного шифрования. Это связано в первую очередь с тем, что известные и широко используемые шифры оперируют блоками данных достаточно большой длины и, как правило, устойчивы к известным видам анализа. Нам же необходимо рассмотреть такой шифр, который бы поддавался анализу. Более того, необходимо, чтобы анализ такого шифра можно было бы провести за время, отведенное в учебном процессе для лабораторных работ. Конечно, в таком случае рассматриваемый шифр оказывается нестойким и по сути дела непригодным для использования его в целях сокрытия данных. Но только так можно рассмотреть основные механизмы, используемые в современных видах анализа с тем, чтобы в дальнейшем на практике уметь оценить стойкость используемых шифров. Более того, эти механизмы необходимо знать и учитывать в случае разработки новых алгоритмов шифрования.

4.1.2. Общий вид учебного алгоритма шифрования

Учебный алгоритм шифрования (УАШ) построен по схеме Фейстеля и содержит в себе три раунда шифрования. Он был разработан авторами учебного пособия для изучения линейного и дифференциального криптоанализа [13]. Общий вид алгоритма представлен на рис. 35.

Как видно из рис. 35, такая организация алгоритма шифрования повторяет структуру алгоритма DES за тем исключением, что здесь сокращено количество раундов шифрования и отсутствуют начальная и конечная перестановки. Отсутствие перестановок объясняется тем, что они по своей сути фактически не влияют на криптографическую стойкость преобразуемых данных. Их наличие в УАШ всего лишь усложнило бы понимание механизмов анализа. Далее будет показано, что функция F, осуществляющая главное криптографическое преобразование, в

своем составе содержит операции, по своей структуре аналогичные операциям, входящим в состав функции F алгоритма шифрования DES. Именно поэтому такой алгоритм можно назвать DES-подобным алгоритмом шифрования.

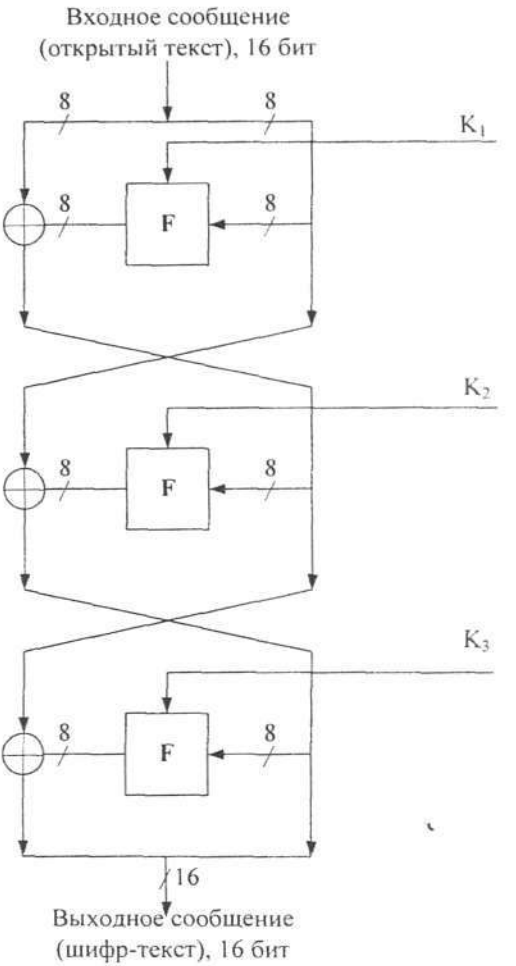


Рис. 35. Общий вид учебного алгоритма шифрования

Рассмотрим более подробно алгоритм шифрования, представленный на рис. 35. Исходное сообщение, которое подлежит преобразованию с помощью данного УАШ, представляет собой последовательность из 16 бит, которые разделяются на две части. Правая часть сообщения преобразуется с помощью функции F под воздействием раундового ключа, после чего складывается по модулю 2 с левой частью, и части сообщения меняются местами. Алгоритм состоит из трех таких раундов, при этом в последнем раунде правая и левая части не меняются местами, а остаются на своих местах, образуя выходное шифрованное сообщение.

4.1.3. Функция F

Теперь перейдем к рассмотрению преобразования, выполняемого с помощью функции F (рис. 36). На вход функции F поступает 8 бит (половина от преобразуемого блока данных). Они проходят через блок перестановки с расширением так, что на выходе мы получаем 12 бит. Перестановку с расширением иногда называют Е-перестановкой (от англ. *expansion*). Расширение происходит путем дублирования некоторых позиций исходного 8-битового сообщения. Данные таблицы перестановки с расширением приведены в табл. 22. Сразу следует оговориться и отметить, что наполнение таблицы перестановки с расширением, представленное в табл. 22, не является фиксированным. Для каждого варианта лабораторных используется своя таблица перестановки с расширением, что позволяет обеспечить каждого обучающегося индивидуальным вариантом.

После прохождения через перестановку с расширением вновь образованное 12-битовое сообщение складывается по модулю 2 с секретным раундовым 12-битовым подключом, выработанным из исходного секретного ключа.

Результат сложения разбивается на три группы по четыре бита, каждая из которых проходит соответственно через блоки замены: Блок 1, Блок 2 и Блок 3. Блоки замены также иногда обозначают как S-блоки (от англ. *substitution*). При этом на выходе первых двух блоков замены образуется три бита, а на выходе третьего блока – два.

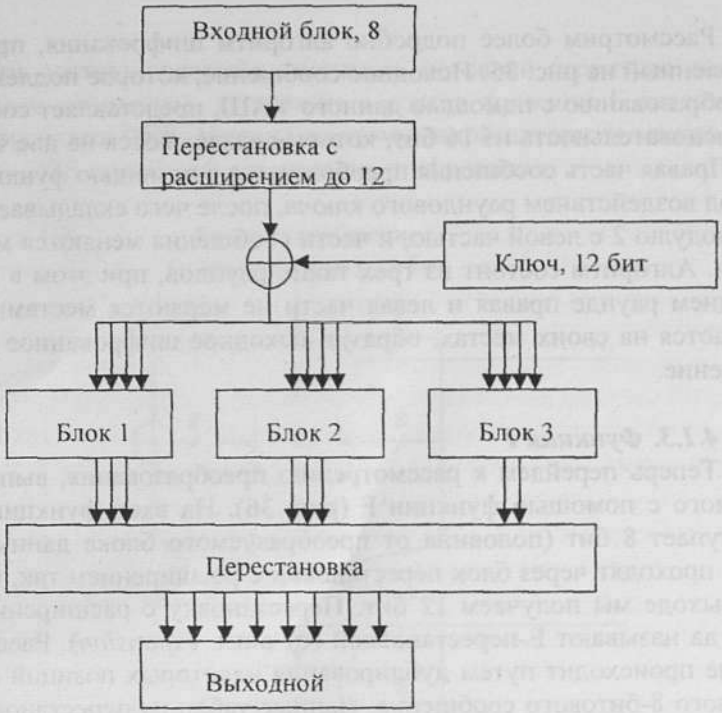


Рис. 36. Функция F

Замена в первых двух блоках производится по приведенным таблицам замены (соответственно табл. 23 и 24) по следующему принципу. Пусть на вход блока замены поступают четыре бита: a1, a2, a3, a4. При этом первый бит определяет номер строки (если a1 = 0, то это соответствует первой строке, а если a1 = 1 – второй), а остальные три – номер столбца (000 = = первому столбцу, 111 = восьмому столбцу) в таблице замены. Результат на пересечении строки и столбца образует выходное значение блока замены.

Несколько иначе дело обстоит с Блоком 3. Из него в отличие от первых двух выходит не три бита, а два. Замена осуществляется согласно табл. 25. При этом, если a1, a2, a3, a4 – входные биты Блока 3, то биты a1 и a4 определяют номер сроки таблицы замены, а биты a2 и a3 – номер столбца (подобно тому,

как происходит замена данных с помощью S-блоков алгоритма DES).

Здесь также стоит заметить, что наполнение трех блоков замены не является фиксированным и будет различаться для каждого индивидуального варианта выполняемой работы.

Таблица 22

Таблица перестановки с расширением

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

Таблица 23

Таблица замены в Блоке 1

a2a3a4 / a1	000	001	010	011	100	101	110	111
0	4	6	1	3	5	7	2	5
1	5	7	2	4	6	1	3	6

Таблица 24

Таблица замены в блоке 2

a2a3a4 / a1	000	001	010	011	100	101	110	111
0	3	5	7	2	4	6	1	7
1	4	6	1	3	5	7	2	1

Таблица 25

Таблица замены в блоке 3

a2a3 / a1 a4	00	01	10	11
00	1	3	2	1
01	2	1	3	2
10	3	2	1	3
11	1	3	2	1

На выходе из блоков замены образуется 8-битовое сообщение, которое претерпевает перестановку согласно табл. 26. Подобные перестановки также называют P-перестановками (от англ. *permutation*). Результат работы перестановки образует 8-

битное сообщение, которое появляется на выходе функции преобразования F.

Таблица 26

Таблица перестановки

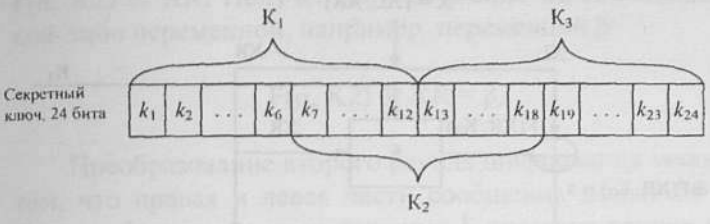
8	7	3	2	5	4	1	6

Выход функции F складывается по модулю 2 с левой частью сообщения, образуя новую левую часть текста. Правая часть сообщения остается неизменной (т. е. такой, какая поступала на вход функции F данного раунда). После этого правая и левая части меняются местами и поступают на вход второго раунда шифрования. Важно отметить, что в последнем раунде правая и левая части не меняются местами. Это сделано для того, чтобы зашифрованные тексты можно было расшифровать с использованием этого же алгоритма шифрования.

4.1.4. Использование раундовых подключей

В каждом раунде при преобразовании данных с помощью функции F происходит сложение данных с секретным раундовым подключом. Каждый раундовый подключ (их всего три – по количеству раундов в алгоритме шифрования) имеет длину 12 битов и вырабатывается из исходного 24-битового ключа по схеме, отображенной на рис. 37.

Как видно из рис. 37, исходный ключ имеет длину 24 бита, которые пронумерованы слева направо от 1 до 24 (то есть бит k1 – самый старший, а бит k24 – самый младший). Таким образом, первые 12 битов с k1 до k12 образуют первый подключ K1, который используется в первом раунде шифрования. Двенадцать битов с k7 до k18 образуют второй подключ K2, который используется во втором раунде шифрования. И, наконец, последние 12 битов с k13 до k24 образуют третий подключ K3, который используется в последнем раунде шифрования.



K_i – секретный подключ i-го раунда
k_j – j-й бит исходного секретного ключа

Рис. 37. Выработка раундового подключа

4.1.5. Использование одного и того же алгоритма для шифрования и расшифрования данных

Как уже отмечалось ранее, один и тот же симметричный алгоритм блочного шифрования, построенный по схеме Фейстеля, может использоваться как для шифрования данных, так и для их расшифрования. Рассмотрим более подробно за счет каких механизмов это становится возможным.

На рис. 38 представлены раунды УАШ и приведено описание преобразования входного сообщения X при его прохождении через раунды шифрования. На вход алгоритма шифрования поступает сообщение X, которое необходимо зашифровать. Так как алгоритм построен по схеме Фейстеля, то происходит разделение исходного сообщения X на две части: левую часть сообщения XL и правую часть сообщения XR. Правая часть сообщения, т. е. значение XR, поступает на вход функции F первого раунда шифрования, где шифруется с использованием подключа K1. Значение, полученное в результате прохождения XR через функцию F, обозначим как F(XR, K1). Выход функции F первого раунда шифрования складывается по модулю 2 с левой частью исходного сообщения XL: $F(XR, K1) \oplus XL$. Полученное выражение для удобства дальнейшей работы можно обозначить какой-либо переменной, например, переменной α:

$$F(XR, K1) \oplus XL = \alpha. \qquad (1)$$

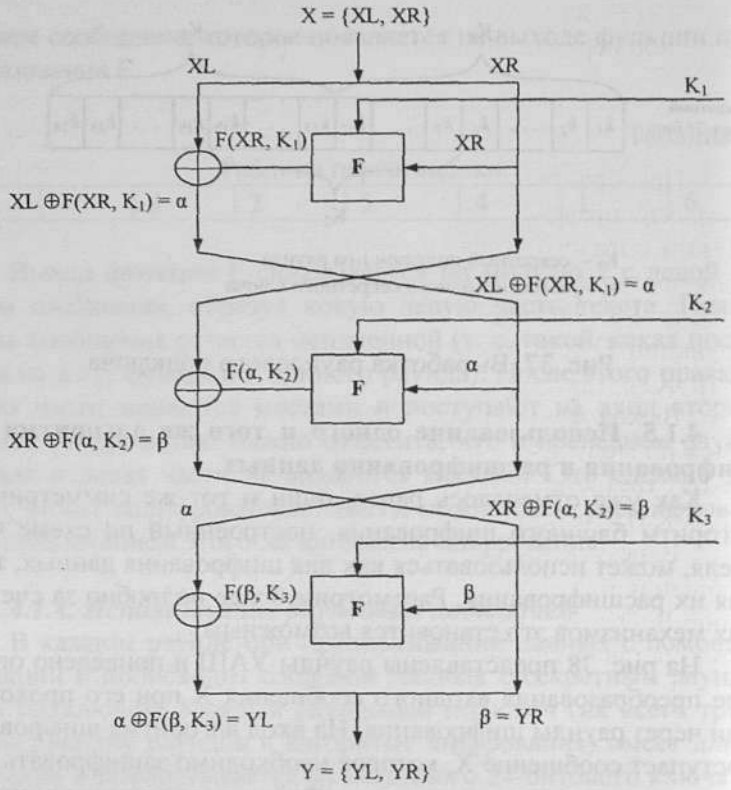


Рис. 38. Преобразование сообщения при шифровании

Преобразование первого раунда шифрования заканчивается тем, что правая и левая части сообщения меняются местами. Таким образом, значение α поступает на вход функции F второго раунда шифрования, где оно шифруется с использованием ключа K2. В результате такого преобразования на выходе функции F второго раунда шифрования образуется значение F(α, K2).

Выход функции F второго раунда шифрования складывается по модулю 2 с левой частью сообщения, поступившей на вход второго раунда шифрования, т. е. со значением XR:

$F(\alpha, K2) \oplus XR$. Полученное выражение можно обозначить какой-либо переменной, например, переменной β:

$$F(\alpha, K2) \oplus XR = \beta. \quad (2)$$

Преобразование второго раунда шифрования заканчивается тем, что правая и левая части сообщения меняются местами. Таким образом, на вход функции F третьего раунда шифрования поступает значение β, где оно шифруется с использованием ключа K3. В результате такого преобразования, на выходе функции F третьего раунда шифрования образуется значение $F(\beta, K3)$.

Выход функции F третьего раунда шифрования складывается по модулю 2 с левой частью сообщения, поступившего на вход третьего раунда шифрования, т. е. со значением α: $F(\beta, K3) \oplus \alpha$.

В результате шифрования левая часть шифрованного сообщения будет равна значению $F(\beta, K3) \oplus \alpha$, а правая – значению β. Эти части сообщения, обозначенные как YL, YR, и будут составлять шифр-текст:

$$F(\beta, K3) \oplus \alpha = YL; \quad (3)$$
$$\beta = F(\alpha, K2) \oplus XR = YR; \quad (4)$$
$$Y = \{YL, YR\}. \quad (5)$$

Теперь рассмотрим обратный процесс, т. е. процесс, при котором из шифрованного сообщения Y будет восстановлено исходное сообщение X (рис. 39). Известно, что при расшифровании необходимо использовать тот же алгоритм, что был использован при шифровании данных. Отличие заключается только в том, что раундовые подключи необходимо использовать в обратном порядке. То есть при расшифровании в первом раунде будет использован подключ K3, а в последнем – подключ K1.

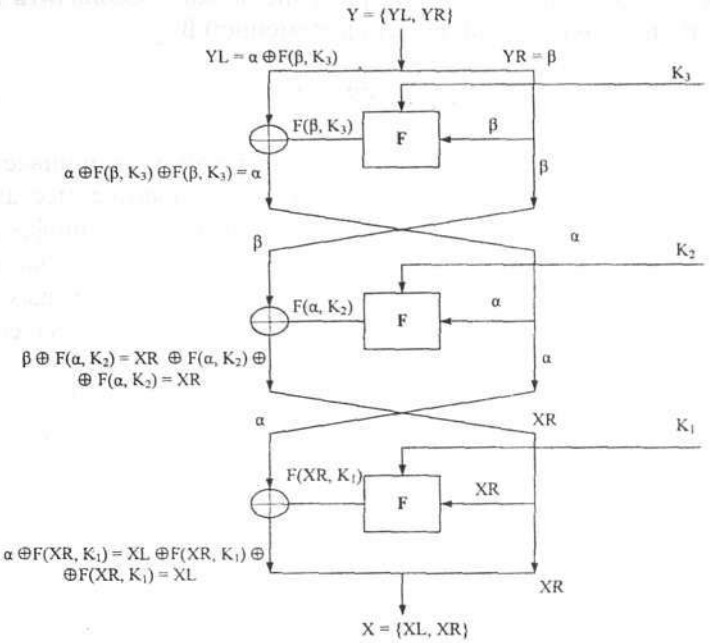


Рис. 39. Преобразование сообщения при расшифровании

При расшифровании на вход алгоритма поступает зашифрованный текст Y, который состоит из двух половинок: Y = {YL, YR}. Правая часть этого сообщения, т. е. значение YR = β, поступает на вход функции F первого раунда, где шифруется с помощью подключа К3. В результате на выходе функции F первого раунда образуется значение F (β, К3).

Выход функции F первого раунда шифрования складывается по модулю 2 с левой частью сообщения, поступившего на вход первого раунда шифрования, т. е. со значением YL. В результате образуется сумма: YL ⊕ F(β, К3). Заменим значение YL в соответствии с формулой (3):

$$YL \oplus F(\beta, K3) = F(\beta, K3) \oplus \alpha \oplus F(\beta, K3) = \alpha.$$

Получается, что в результате сложения по модулю 2 левой части входного сообщения первого раунда и выхода функции F

первого раунда образуется значение α (два одинаковых значения F(β, K3) в результате сложения по модулю 2 образуют ноль). После этого правая и левая части сообщения меняются местами. Таким образом, на вход второго раунда поступает сообщение, левая часть которого равна β, а правая – α.

После прохождения через функцию F второго раунда значение α преобразуется в значение F(α, K2), так как во втором раунде происходит шифрование с использованием раундового подключа K2.

Выход функции F второго раунда шифрования складывается по модулю 2 с левой частью сообщения, поступившей на вход второго раунда, т. е. со значением β. В результате образуется сумма: $\beta \oplus F(\alpha, K2)$. Заменим значение β в соответствии с формулой (2):

$$\beta \oplus F(\alpha, K2) = F(\alpha, K2) \oplus XR \oplus F(\alpha, K2) = XR.$$

Таким образом, в результате сложения по модулю 2 левой части входного сообщения второго раунда и выхода функции F второго раунда образуется значение XR. После этого правая и левая части сообщения меняются местами. Так, на вход третьего раунда поступает сообщение, левая часть которого равна α, а правая – XR.

В заключение на вход функции F третьего раунда поступает значение XR, которое под действием раундового подключа K1 преобразуется в значение F(XR, K1).

Выход функции F последнего раунда складывается по модулю 2 с левой частью сообщения, поступившей на вход третьего раунда, т. е. со значением α. В результате образуется сумма: $\alpha \oplus F(XR, K1)$. Заменим значение α в соответствии с формулой (1):

$$\alpha \oplus F(XR, K1) = F(XR, K1) \oplus XL \oplus F(XR, K1) = XL$$

В результате сложения по модулю 2 левой части входного сообщения первого раунда и выхода функции F первого раунда образуется значение XL.

В итоге левая часть выходного сообщения будет равна значению XL, а правая – значению XR: X = {XL, XR}. Эти части сообщения и будут составлять искомое значение, которое ранее нами было зашифровано.

4.1.6. Пример шифрования данных с использованием УАШ

Для того чтобы лучше освоить принцип шифрования данных с использованием блочных симметричных шифров, рассмотрим пример шифрования данных с помощью описанного выше учебного алгоритма шифрования. Для этого возьмем сообщение X = 5520310 и ключ K = 176061910. Первым делом необходимо перевести эти сообщения в двоичную систему счисления. При переводе необходимо помнить, что сообщение X должно занимать 16 бит, а ключ K – 24 бита. Если в случае перевода в двоичную систему счисления полученная последовательность битов оказывается короче, чем того требует алгоритм шифрования, необходимо дописать недостающее число нулей к старшим значащим разрядам. Таким образом, в результате перевода чисел получим:

X = 1101 0111 1010 0011
K = 0001 1010 1101 1101 0110 1011

Обратите внимание, что в значении ключа K появились три старших нуля. Следующим шагом является определение раундовых подключей в соответствии с рис. 37:

K1 = 0001 1010 1101;
K2 = 1011 0111 0101;
K3 = 1101 0110 1011.

Итак, на вход алгоритма шифрования поступает блок данных из 16 бит X = 1101 0111 1010 0011, который разделяется на две части по 8 бит: левую часть – 1101 0111 и правую часть – 1010 0011. Для наглядности процесс зашифрования данных представлен на рис. 40.

Правая часть сообщения (значение 1010 0011) поступает на вход функции F первого раунда шифрования, где преобразуется с использованием раундового подключа K1 в соответствии с рис. 36.

Первым преобразованием функции F является перестановка с расширением, которая направлена на перемешивание и размножение битов исходного сообщения. Для того чтобы применить перестановку с расширением, необходимо пронумеровать биты исходной последовательности от 1 до 8 слева направо. После этого биты исходной последовательности необходимо преобразовать с помощью табл. 22. Так, в соответствии с табл. 22, третий бит исходной последовательности станет первым, четвертый бит – вторым, первый – бит третьим и т.д. В результате преобразования исходная последовательность 1010 0011 приводится к виду 1010 0101 1100. Для наглядности процесс применения перестановки с расширением изображен на рис. 41.

После перестановки E происходит сложение данных с раундовым подключом K1 по модулю 2. Напомним, что операция сложения по модулю 2 работает следующим образом: если складываются два одинаковых бита (т. е. или два нуля, или две единицы), то в результате сложения образуется 0; если же складываются два разных бита (0 и 1 или 1 и 0), то в результате сложения образуется 1. Итак, в результате сложения по модулю 2 данных после перестановки с расширением и раундового подключа K1 получается значение 1011 1111 0001 (рис. 42).

После сложения с раундовым подключом последовательность из 12 битов разбивается на три группы по четыре бита. Каждая из групп поступает на вход одного из трех блоков замены, где преобразуется в соответствии с табл. 23 – 25. Напомним, что первые два блока замены имеют одинаковый принцип работы. Работа третьего блока несколько отличается от первых. Рассмотрим данные преобразования более подробно. Так, в соответствии с последовательностью, полученной после сложения с раундовым подключом, на вход первого блока замены поступает значение 1011. Первый бит последовательности указывает на номер строки в табл. 23, последние три бита – на номер

столбца. В результате такого преобразования на выходе блока замены образуется значение 100. Для наглядности процесс преобразования данных с помощью первого блока замены представлен на рис. 43. Аналогичным образом на вход второго блока замены поступает значение 1111, которое в соответствии с табл. 24 преобразуется к значению 001.

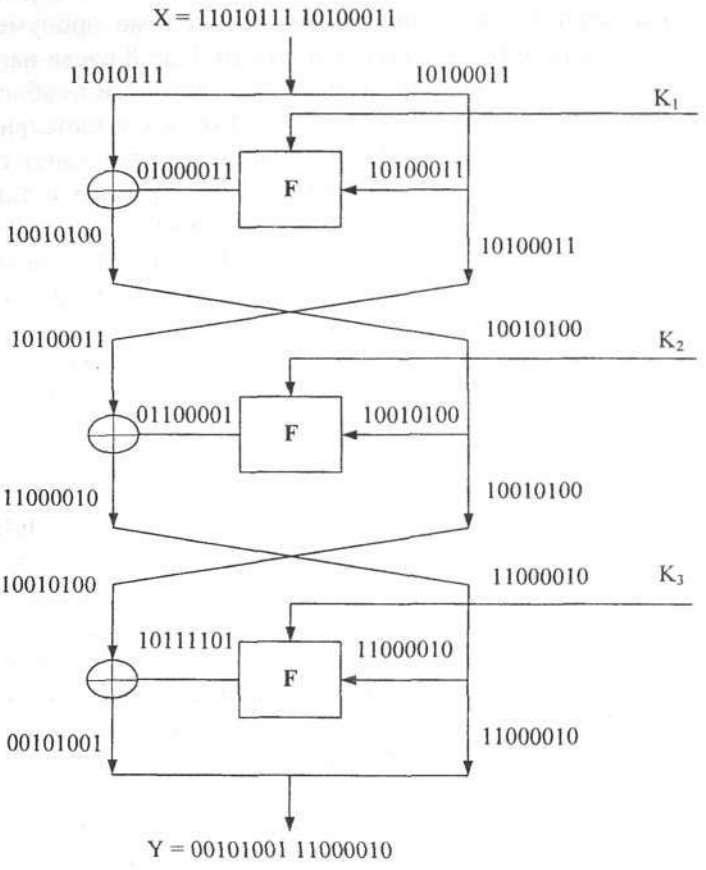


Рис. 40. Процесс шифрования сообщения
X = 1101 0111 1010 0011 с помощью секретного ключа
K = 0001 1010 1101 1101 0110 1011

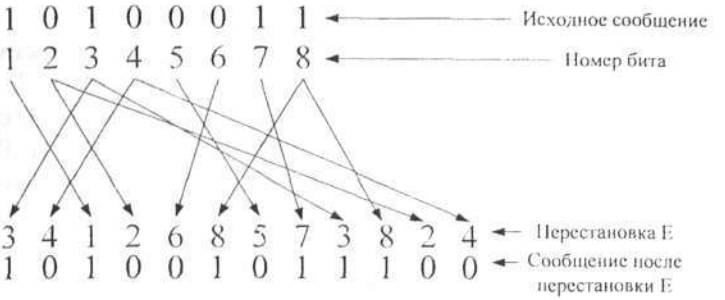


Рис. 41. Применение операции перестановки с расширением

Рис. 42. Сложение данных после перестановки с расширением с раундовым подключом K1

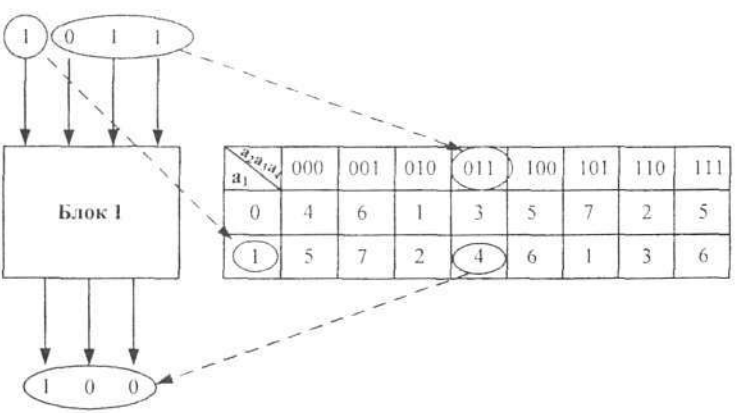


Рис. 43. Преобразование данных с помощью первого блока замены

Последние четыре бита последовательности – 0001 – преобразуются с помощью третьего блока замены в соответствии с табл. 25. Здесь первый и последний биты входа в Блок 3, т. е. значение 01, образуют номер строки в таблице замены, а средние два бита, т. е. значение 00, образуют номер столбца. В результате преобразования на выходе блока замены образуется значение 10 (обратим внимание, что на выходе последнего блока замены образуется два бита в отличие от первых двух блоков, где на выходе были трехбитовые значения). Для наглядности процесс преобразования данных с использованием последнего блока замены представлен на рис. 44.

Таким образом, после применения преобразования с помощью блоков замены на выходе образуется последовательность из 8 бит: 100 001 10.

Последним преобразованием функции F является обычная перестановка, работа которой осуществляется в соответствии с табл. 26 и аналогична работе перестановки с расширением, которая была описана раньше. В результате перестановки последовательность 10000110 преобразуется в последовательность 01000011, которая и является выходом функции F первого раунда шифрования (см. рис. 40).

После того как выполнено преобразование с помощью функции F, необходимо сложить левую часть сообщения на входе первого раунда шифрования (значение 11010111) с выходом функции F первого раунда шифрования (т. е. со значением 01000011). В результате образуется значение 10010100. Далее правая и левая части сообщения меняются местами, т. е. на вход второго раунда шифрования поступает последовательность 10100011 10010100.

Преобразования дальнейших двух раундов схожи с теми, которые были проведены в первом раунде шифрования. Мы не будем здесь повторять подробное описание одних и тех же действий. Однако для того чтобы проследить дальнейшие преобразования, отобразим их с помощью функции F для второго и третьего раундов, представленных соответственно на рис. 45 и 46.

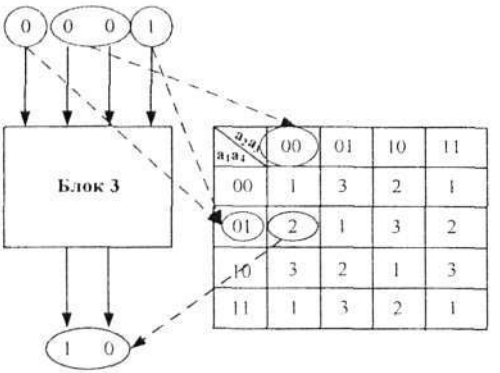


Рис. 44. Преобразование данных с помощью третьего блока замены

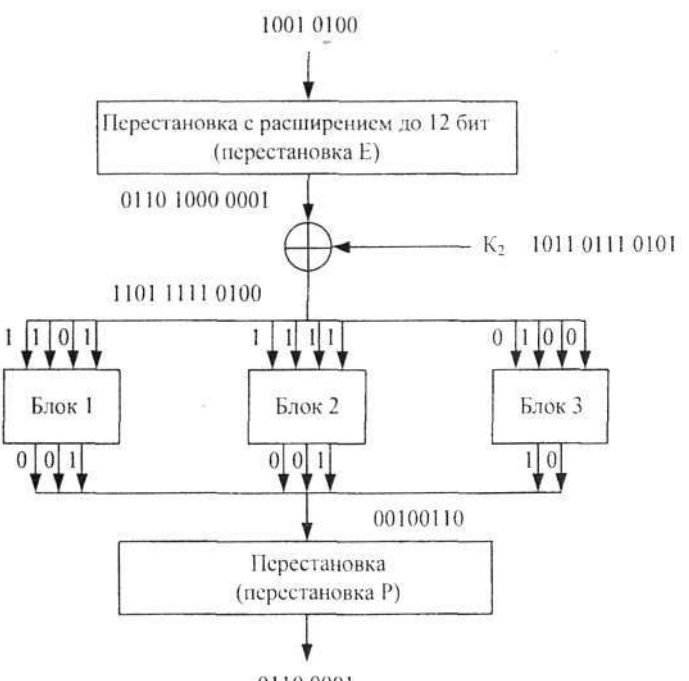


Рис. 45. Преобразование функции F второго раунда

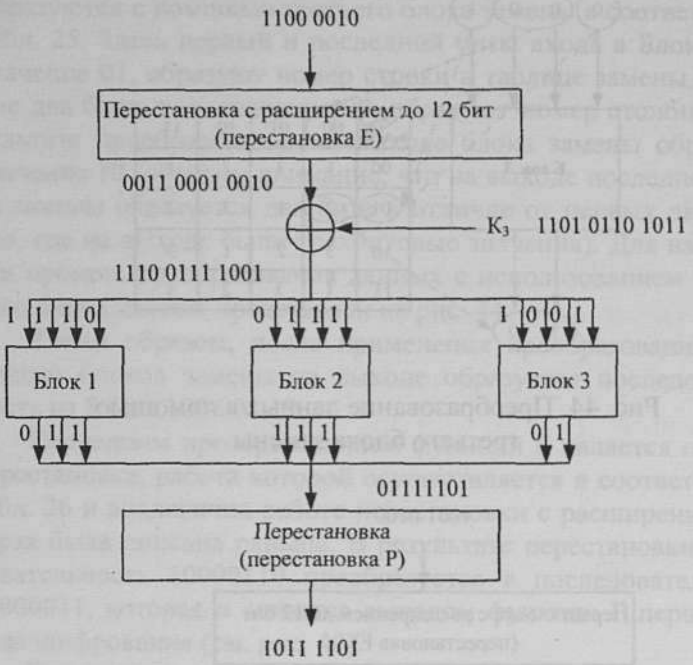


Рис. 46. Преобразование функции F третьего раунда

На вход функции F второго раунда шифрования поступает значение 1001 0100, которое преобразуется к значению 0110 0001 в соответствии с рис. 45. После этого выход функции F второго раунда шифрования (т. е. значение 0110 0001) складывается по модулю 2 с левой частью сообщения на входе второго раунда шифрования (значение 1010 0011). В результате образуется значение 1100 0010. Далее правая и левая части сообщения меняются местами, т. е. на вход третьего раунда шифрования поступает последовательность 10010100 11000010.

Так, на вход функции F третьего раунда шифрования поступает значение 1100 0010, которое преобразуется к значению 1011 1101 в соответствии с рис. 46. После этого выход функции F третьего раунда шифрования (т. е. значение 1011 1101) складывается по модулю 2 с левой частью сообщения на входе третьего раунда шифрования (значение 10010100). В результате

образуется значение 0010 1001. Так как третий раунд шифрования является последним, в нем не происходит обмена данными между правой и левой частями сообщения. Таким образом, на выход третьего раунда шифрования поступает последовательность 00101001 11000010, которая и является искомым шифрованным сообщением.

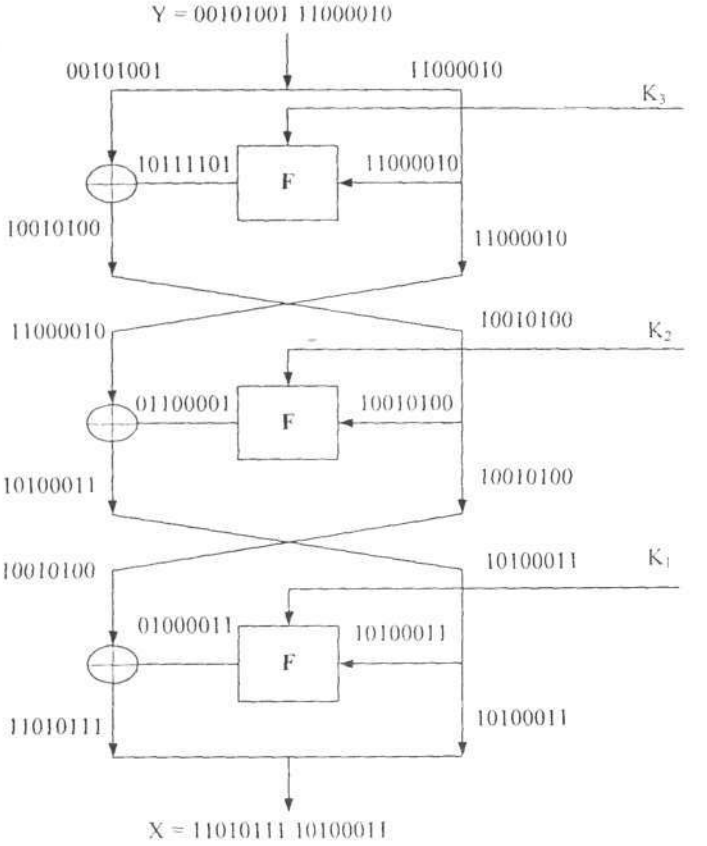


Рис. 47. Процесс расшифрования сообщения Y = 00101001 11000010 с помощью секретного ключа K = 0001 1010 1101 1101 0110 1011

Расшифрование данных проводится по той же схеме, что и шифрование, с той разницей, что раундовые подключи используются в обратном порядке. Процесс расшифрования сообщения 00101001 11000010 представлен на рис. 47. Вы можете выполнить его самостоятельно, после чего сверить все промежуточные значения.

4.2. Упрощенный алгоритм шифрования DES (S-DES)

Упрощенный DES – это алгоритм шифрования, имеющий, скорее, учебное, чем практическое значение. По свойствам и структуре он подобен DES, но имеет гораздо меньше параметров. Данный алгоритм был разработан профессором Эдвардом Шейфером из университета Санта-Клара. Для удобства дальнейшей работы будем называть данный алгоритм S-DES [6].

Алгоритм шифрования S-DES представляет собой симметричный блочный шифр, преобразующий блок из 8 битов под воздействием 10-битового ключа. Работа алгоритма начинается с начальной перестановки IP, после этого выполняются два раунда по классической схеме Фейстеля. 8-битовый блок разделяется на две половины по 4 бита. В каждом раунде правая половина проходит через преобразование функции F (где все элементы подобны элементам алгоритма DES) и преобразуется под воздействием раундового подключа K_i. Выход функции F складывается по модулю 2 с левой частью сообщения, после чего левая и правая части сообщения меняются местами. Во втором раунде левая и правая половины сообщения местами не меняются. Завершается работа алгоритма примененим конечной перестановки IP^{-1}.

В каждом раунде используется подключ K_i, который вырабатывается из исходного 10-битового секретного ключа K. То есть всего используется два раундовых подключа K_1 и K_2.

Процесс расшифрования аналогичен процессу зашифрования за тем исключением, что раундовые подключи применяются в обратном порядке, в первом раунде применяется подключ K_2, во втором – подключ K_1.

Теперь давайте рассмотрим элементы алгоритма S-DES более подробно.

4.2.1. Вычисление ключей S-DES

В алгоритме S-DES используется 10-битовый ключ, который должен быть как у отправителя, так и у получателя сообщения. Из этого ключа на определенных этапах шифрования и расшифрования генерируется два 8-битных подключа. На рис. 48 показана схема процедуры создания этих подключей.

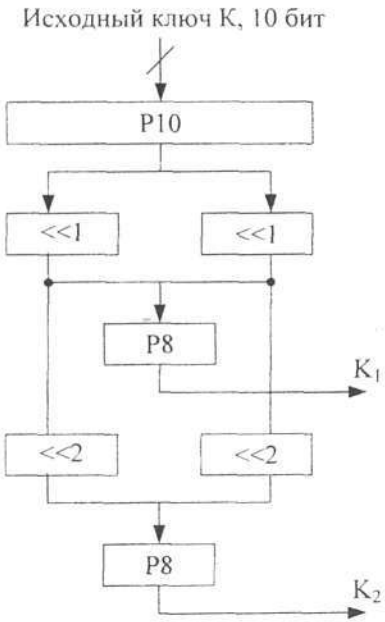


Рис. 48. Вычисление ключей S-DES

Сначала выполняется перестановка битов ключа следующим образом. Если 10-битовый ключ представить в виде (k_1, k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9,k_{10}), то перестановку P10 можно задать формулой

$$P10(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = \\ = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6).$$

Можно также представить перестановку P10 в табличной форме так, как это показано в табл. 27.

Таблица 27

Перестановка P10 для выработки раундовых подключей

3	5	2	7	4	10	1	9	8	6

Эту таблицу следует читать слева направо. Каждый ее элемент идентифицирует позицию бита исходных данных в генерируемой выходной последовательности. Иными словами, первым битом в выходной последовательности будет третий бит исходной последовательности, вторым – пятый и т.д. Например, в соответствии с данной таблицей ключ (1010000010) будет преобразован к виду (1000001100). После этого отдельно для первых пяти битов и отдельно для вторых выполняется циклический сдвиг влево (<<1). В нашем случае в результате будет получена последовательность (00001 11000).

Затем применяется перестановка со сжатием P8, в результате которой из 10-битового ключа сначала выбираются, а затем переставляются 8 битов в соответствии с табл. 28.

Таблица 28

Престановка со сжатием P8

6	3	7	4	8	5	10	9

В результате этой операции получается подключ (K_1). В нашем примере он будет иметь вид (10100100).

Теперь нужно вернуться к двум 5-битовым строкам, полученным в результате применения циклического сдвига влево на одну позицию, и выполнить с каждой из этих строк циклический сдвиг влево еще на две позиции (<<2). В нашем конкретном случае значение (00001 11000) будет преобразовано к виду (00100 00011). Наконец, применив к полученной в результате последовательности перестановку P8, получим подключ K_2. Для нашего примера результатом будет (01000011).

Для наглядности процесс вычисления раундовых подключей приведен на рис. 49.

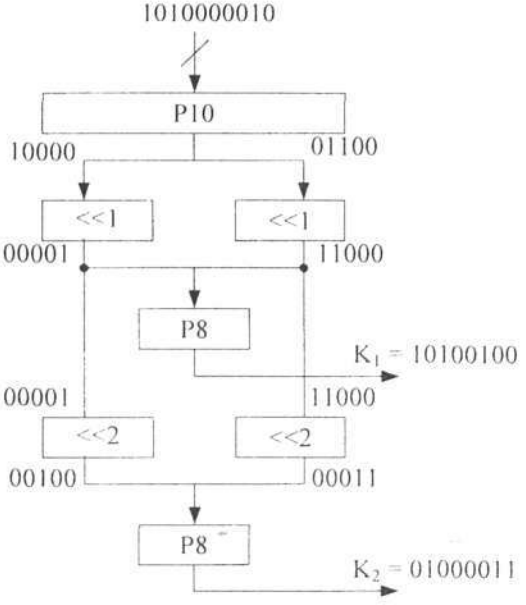


Рис. 49. Пример вычисленияраундовых подключей для алгоритма S-DES

4.2.2. Шифрование S-DES

На рис. 50 представлена схема алгоритма шифрования S-DES. Как уже упоминалось, процесс шифрования представляет собой начальную перестановку, выполнение двух раундов преобразования по схеме Фейстеля и конечную перестановку.

Начальная и конечная перестановки

На вход алгоритма поступает 8-битовый блок открытого текста, к которому применяется начальная перестановка, заданная функцией IP в соответствии с табл. 29.

Таблица 29

Начальная перестановка IP

2	6	3	1	4	8	5	7

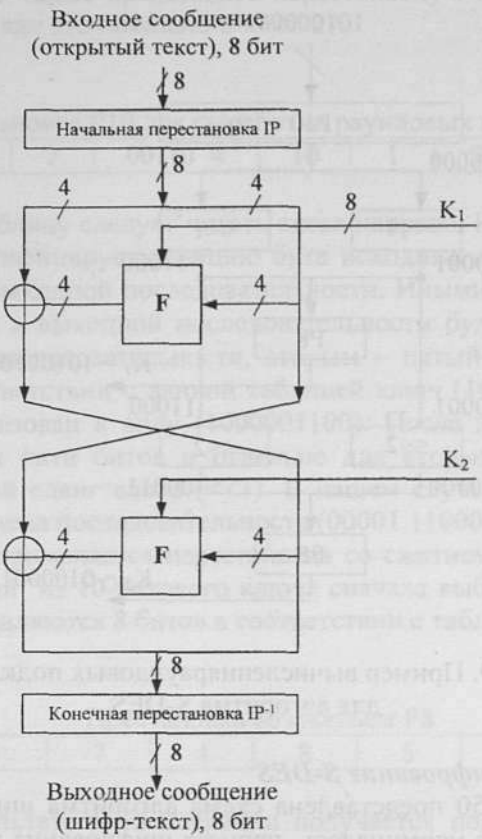


Рис. 50. Общий вид алгоритма шифрования S-DES

Все 8 бит открытого текста сохраняют свои значения, но меняется порядок их следования. На завершающей стадии алгоритма выполняется конечная перестановка в соответствии с табл. 30.

Таблица 30

Конечная перестановка IP^{-1}

4	1	3	5	7	2	8	6

Как легко убедиться с помощью простой проверки, данные перестановки являются взаимнообратными, то есть для них выполняются следующие равенства:

$$IP^{-1}(IP(X)) = X;$$
$$IP\,(IP^{-1}(X)) = X.$$

Функция F

Самым сложным компонентом S-DES является функция F, представляющая собой комбинацию перестановок и подстановок (рис. 51).

На вход функции F поступает 4 бита (половина от преобразуемого блока данных). Считается, что преобразуемый блок данных условно можно разделить на левую XL и правую XR половины, поэтому можно сказать, что на вход функции F поступают 4 бита XR. Они проходят через блок перестановки с расширением так, что на выходе образуется 8 бит. Расширение происходит путем дублирования каждой позиции исходного 4-битового сообщения в соответствии с табл. 31. После прохождения через перестановку с расширением вновь образованное 8-битовое сообщение складывается по модулю 2 с секретным раундовым 8-битовым подключом K_i, выработанным из исходного секретного ключа.

Таблица 31

Перестановка с расширением E/Pв функции F

4	1	2	3	2	3	4	1

Результат сложения разбивается на две группы по 4 бита, каждая из которых проходит соответственно через блоки замены: S_1 и S_2. Замена осуществляется так же, как и для Блока 3 в алгоритме УАШ. Заполнения таблиц для блоков S_1 и S_2 представлены соответственно в табл. 32 и 33. При этом, если a1, a2, a3, a4 – входные биты Блока 3, то биты a1 и a4 определяют номер сроки таблицы замены, а биты a2 и a3 – номер столбца (подобно тому, как происходит замена данных с помощью S-блоков алгоритма DES).

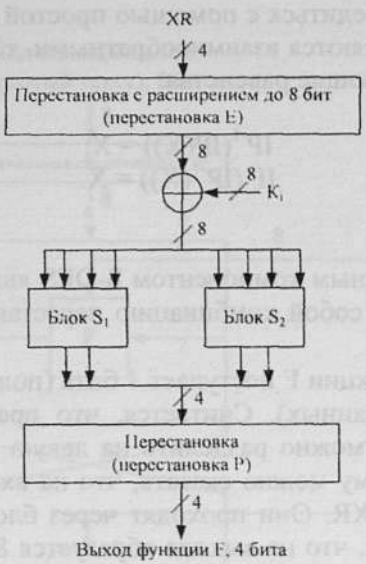


Рис. 51. Функция F для алгоритма шифрования S-DES

Таблица 32

Таблица замены в блоке S_1

a2a3 / a1 a4	00	01	10	11
00	1	0	3	2
01	3	2	1	0
10	0	2	1	3
11	3	1	3	1

Таблица 33

Таблица замены в блоке S_2

a2a3 / a1 a4	00	01	10	11
00	1	1	2	3
01	2	0	1	3
10	3	0	1	0
11	2	1	0	3

На выходе из блоков замены образуется 4-битовое сообщение, которое претерпевает перестановку P согласно табл. 34. Результат работы перестановки образует 4-битовое сообщение, которое появляется на выходе функции преобразования F.

Таблица 34

Перестановка P для функции F

2	4	3	1

4.2.3. Пример шифрования данных с использованием алгоритма S-DES

Для того чтобы лучше освоить принцип шифрования данных с использованием алгоритма S-DES, рассмотрим пример. Для этого возьмем сообщение X = 35. В качестве подключей будем использовать значения K1 и K2, полученные ранее в подразд. 4.2.1. Первым делом необходимо перевести значение X в двоичную систему счисления: X = 00100011.

Итак, на вход алгоритма шифрования поступает блок данных из 8 бит X = 0010 0011, который разделяется на две части по 4 бита: левую часть – 001 и правую часть –0011. Для наглядности процесс зашифрования данных представлен на рис. 52.

Первым преобразованием является начальная перестановка IP, которая выполняется в соответствии с табл. 29. Биты исходной последовательности 0010 0011 нумеруются слева направо от 1 до 8 и подвергаются перестановке: второй бит перемещается в первую позицию, шестой – во вторую и т.д. В итоге значение 0010 0011 преобразуется к виду 0010 0101.

Правая часть сообщения (значение 0101) поступает на вход функции F первого раунда шифрования, где преобразуется с использованием раундового подключа K_1 в соответствии с рис. 53.

Первым преобразованием функции F является перестановка с расширением, которая направлена на перемешивание и размножение битов исходного сообщения. Для того чтобы применить перестановку с расширением, необходимо пронумеровать биты исходной последовательности от 1 до 4 слева направо. После этого биты исходной последовательности необходи-

мо преобразовать с помощью табл. 31. Так, в соответствии с табл. 31, четвертый бит исходной последовательности станет первым, первый бит – вторым, второй – бит третьим и т.д. В результате преобразования исходная последовательность 0101 приводится к виду 1010 1010. Для наглядности процесс применения перестановки с расширением изображен на рис. 54.

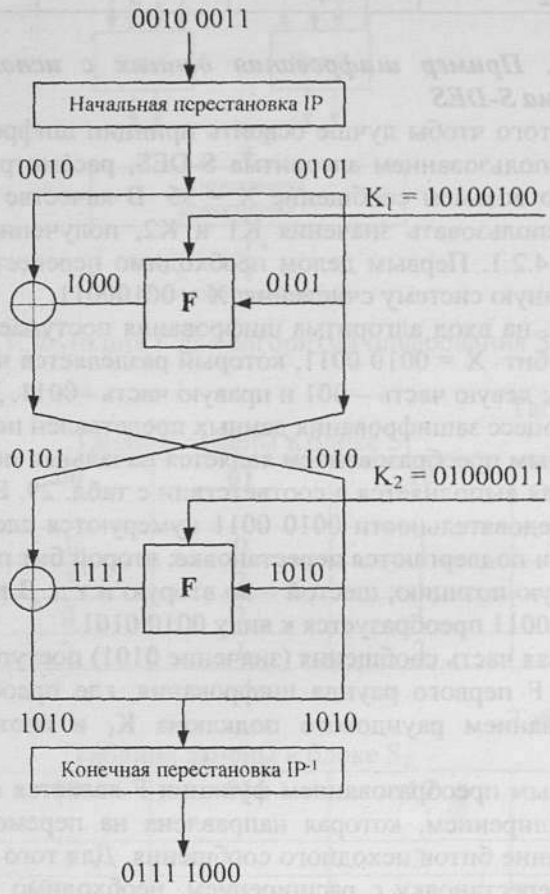


Рис. 52. Процесс шифрования сообщения $X = 35_{10}$ с помощью алгоритма шифрования S-DES ($K = 642_{10}$)

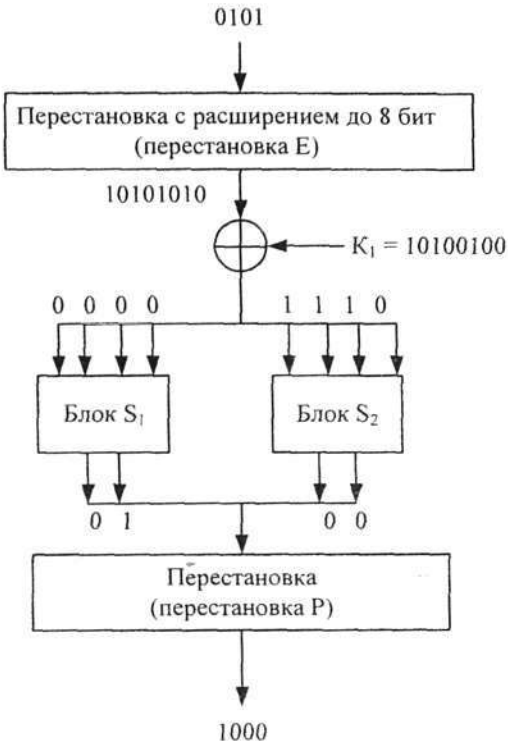


Рис. 53. Преобразование функции F первого раунда

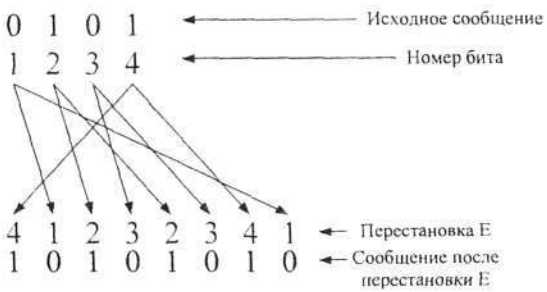


Рис. 54. Применение операции перестановки с расширением

После перестановки E происходит сложение данных с раундовым подключом K_1 по модулю 2. В результате сложения по модулю 2 данных после перестановки с расширением и раундового подключа K_1 получается значение 0000 1110 (рис. 55).

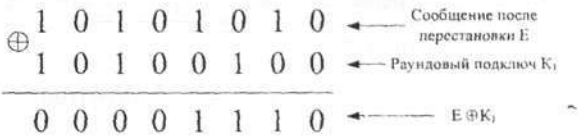


Рис. 55. Сложение данных после перестановки с расширением с раундовым подключом K_1

После сложения с раундовым подключом последовательность из 8 битов разбивается на две группы по 4 бита. Каждая из групп поступает на вход блоков замены S_1 и S_2, где преобразуется в соответствии с табл. 32 и 33. Напомним, что первый и последний биты входа в S-блок образуют номер строки в таблице замены, а средние 2 бита образуют номер столбца. Так, на вход блока S_1 поступает значение 0000, а это значит, что в табл. 32 мы выбираем элемент на пересечении строки 00 и столбца 00 – это значение 1 (так как на выходе S-блока 2 бита, то записываем его как 01). На вход блока S_2 поступает значение 1110, а это значит, что в табл. 33 мы выбираем элемент на пересечении строки 10 и столбца 11 – это значение 1 (записываем его как 00). Таким образом, после применения преобразования с помощью блоков замены на выходе образуется последовательность из 4 бит: 0100.

Последним преобразованием функции F является обычная перестановка, работа которой осуществляется в соответствии с табл. 34 и аналогична работе перестановки с расширением, которая была описана раньше. В результате перестановки последовательность 0100 преобразуется в последовательность 1000, которая и является выходом функции F первого раунда шифрования (см. рис. 52).

После того как выполнено преобразование с помощью функции F, необходимо сложить левую часть сообщения на входе первого раунда шифрования (значение 0010) с выходом

функции F первого раунда шифрования (со значением 1000). В результате образуется значение 1010. Далее правая и левая части сообщения меняются местами, т. е. на вход второго раунда шифрования поступает последовательность 0101 1010.

Преобразования второго раунда выполняются аналогично первому. На его вход поступает значение 1010. Во втором раунде в функции F используется второй раундовый подключ K_2. Преобразования происходят так, как показано на рис. 56.

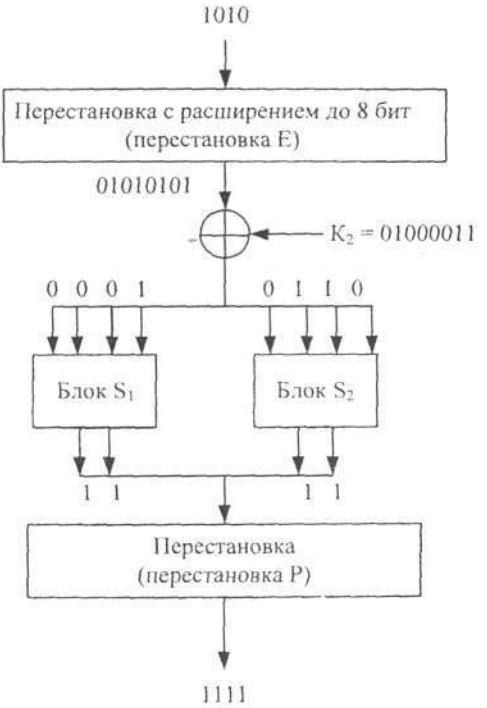


Рис. 56. Преобразование функции F второго раунда

В соответствии с рис. 56, на выходе функции F второго раунда шифрования образуется значение 1111. Это значение складывается по модулю 2 с левой частью сообщения на входе второго раунда, т. е. со значением 0101. В результате получает-

ся 1010. Таким образом, после двух раундов преобразования (после второго раунда левая и правая половины текста местами не меняются) образуется значение 1010 1010. К нему применяется конечная перестановка IP^{-1} в соответствии с табл. 30. В результате образуется значение 0111 1000, которое является шифр-текстом.

Расшифрование данных проводится по той же схеме, что и зашифрование, с той разницей, что раундовые подключи используются в обратном порядке. Процесс расшифрования сообщения 0111 1000 представлен на рис. 57. Вы можете выполнить его самостоятельно, после чего сверить все промежуточные значения.

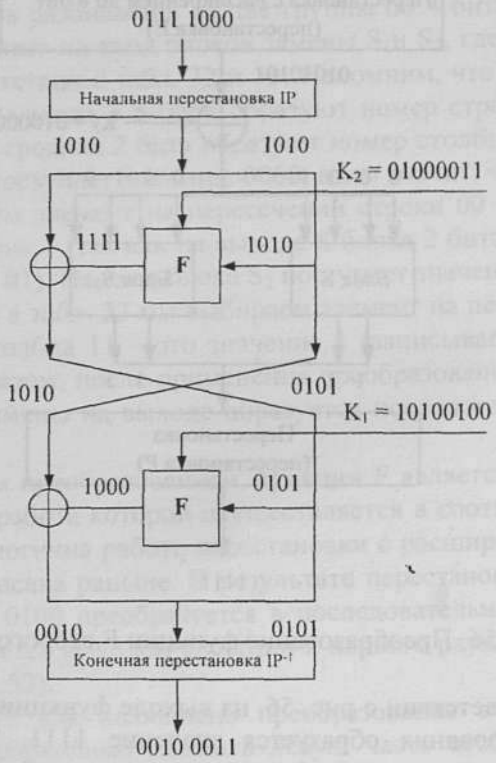


Рис. 57. Процесс расшифрования сообщения Y = 0111 1000 с помощью алгоритма шифрования S-DES ($K = 642_{10}$)

4.3. Алгоритм шифрования S_AES

4.3.1. Математическое обоснование алгоритма

Алгоритм шифрования Simple AES или S_AES разработан в учебных целях и предназначен для обучения будущих криптографов и криптоаналитиков. В основу алгоритма S_AES заложены те же основные принципы, что и в новом стандарте шифрования США AES. Данный алгоритм оперирует полубайтами, которые рассматриваются как элементы поля $GF(2^4)$.

Элементами поля $GF(2^4)$ являются многочлены степени не более трех, которые могут быть заданы строкой своих коэффициентов. Если представить байт в виде

$$\{a_3, a_2, a_1, a_0\}, \text{ где } a_i \in \{0, 1\}, \text{ i меняется от 0 до 3.}$$

Например, полубайту {0101} (или {5} в шестнадцатеричной системе счисления) соответствует многочлен $x^2 \oplus 1$.

Операция сложения над элементами поля представляет собой поразрядное сложение по модулю 2. Умножение в поле $GF(2^4)$ представляет собой операцию умножения со взятием результата по модулю неприводимого многочлена четвертой степени. Для алгоритма S_AES мы взяли многочлен $\varphi(x) = x^4 \oplus x \oplus 1$.

Раундовые преобразования в S_AES оперируют байтами. Байту может быть поставлен в соответствие многочлен a(x) с коэффициентами из $GF(2^4)$ степени не более единицы:

$$a(x) = a_1 x \oplus a_0.$$

При сложении двух многочленов с коэффициентами из $GF(2^4)$ получаем

$$a(x) \oplus b(x) = (a_1 \oplus b_1)x \oplus (a_0 \oplus b_0).$$

Если же мы хотим перемножить два многочлена $a(x) = a_1 x \oplus a_0$ и $b(x) = b_1 x \oplus b_0$, то в результате получится многочлен

$$c(x) = a(x) \otimes b(x) = (a_1x \oplus a_0) \otimes (b_1x \oplus b_0) =$$
$$= a_1b_1x^2 \oplus a_0b_1x \oplus b_0a_1x \oplus a_0b_0 = a_1b_1x^2 \oplus (a_0b_1 \oplus b_0a_1)x \oplus a_0b_0.$$

Для того чтобы результат умножения мог быть представлен байтом, необходимо взять результат по модулю многочлена степени не более двух. Нами был взят многочлен $x^2 \oplus 1$. Таким образом, результатом $c(x)$ умножения $\otimes$ двух многочленов по модулю $x^2 \oplus 1$

$$c(x) = a(x) \otimes b(x)$$

будет многочлен

$$c(x) = c_1x \oplus c_0,$$

где

$$c0 = a_0b_0 \oplus a_1b_1,$$
$$c1 = a_1b_0 \oplus a_0b_1.$$

Итак, если записать в матричной форме, то:

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ a_1 & a_0 \end{bmatrix} \bullet \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

Пусть $c(x) = c_1x \oplus c_0$. Умножению на x многочлена $c(x)$ с коэффициентами из $GF(2^4)$ по модулю $x^2 \oplus 1$ соответствует циклический сдвиг полубайтов в пределах байта в сторону старшего полубайта, так как:

$$x \otimes c(x) = c \otimes (c_1x \oplus c_0) = c_1x_2 \oplus c_0x = c_1(x_2 \oplus 1) \oplus c_0x \oplus c_1 =$$
$$= c_0x \oplus c_1.$$

4.3.2. Описание алгоритма S_AES

Алгоритм S_AES оперирует 16-битовыми входными данными и данными секретного ключа такой же длины. Входные данные алгоритма S_AES обозначаются как полубайты состоя-

ния S_{00}, S_{10}, S_{01} и S_{11} и представляют собой массив из двух строк и двух столбцов:

S_{00}	S_{01}
S_{10}	S_{11}

Ключ шифрования состоит из четырех полубайтов K_{00}, K_{10}, K_{01} и K_{11} и также представляется в виде массива из двух строк и двух столбцов:

K_{00}	K_{01}
K_{10}	K_{11}

Алгоритм S_AES состоит из двух раундов. Перед началом первого раунда происходит сложение исходных данных с первым подключом K1.

Всего в алгоритме S_AES используются три подключа. Есть три варианта их использования. В первом случае можно использовать в качестве раундового подключа заранее определенный секретный ключ K, т. е. в этом случае в каждом раунде будет использоваться один и тот же подключ K. Во втором случае можно заранее определить три секретных подключа и использовать их при шифровании. И, наконец, третьим вариантом, пожалуй, самым сложным, но в то же время и самым правильным, является извлечение раундового подключа из исходного секретного ключа K.

Первый раунд состоит из четырех преобразований, первое из которых заключается в замене полубайтов с помощью S-блока. Второе преобразование заключается в сдвиге строк на один полубайт, третье представляет собой перемешивание столбцов с помощью умножения столбцов данных на многочлен первой степени по модулю $x^2 \oplus 1$. И, наконец, последнее четвертое преобразование представляет собой поразрядное сложение данных с раундовым подключом по модулю 2.

Второй раунд алгоритма S_AES содержит те же преобразования, что и первый раунд, за исключением операции перемешивания столбцов. В последнем, втором раунде алгоритма

S_AES, она отсутствует так же, как и в стандарте шифрования AES.

Рассмотрим каждую из вышеперечисленных операций более подробно.

Замена полубайтов (Sub Half-Bytes*())

С помощью операции **SubHalf-Bytes*()** осуществляется нелинейная замена полубайтов данных. Операция **SubHalf-Bytes*()** применяется к каждому полубайту данных независимо от остальных данных Таблица замены S-блока является инвертируемой и построена из следующих двух преобразований входного полубайта:

1) получение обратного элемента относительно умножения (т. е. мультипликативного обратного) в поле $GF(2^4)$;

2) применение преобразования над $GF(2^4)$, определенного следующим образом:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Иными словами, вышеописанное преобразование можно представить в виде уравнения

$$b_i' = b_i \oplus b_{(i+2) \bmod 4} \oplus b_{(i+3) \bmod 4} \oplus c_i,$$

где $c_0 = c_3 = 1$ и $c_1 = c_2 = 0$, а b_i и b_i' – соответственно исходное и преобразованное значение i-го бита.

На рис. 58 показано применение преобразования **SubHalf-Bytes*()** к шифруемым данным. Сам S-блок замены приведен в табл. 35.

При рассмотрении полубайта {xy} x обозначает два старших бита полубайта, а y – два младших. Так, например, полубайт {d} (или {1101} в двоичной системе счисления) будет заменен на полубайт {4}, стоящий на пересечении третьей строки и второго столбца табл. 35.

Таблица 35

S-блок замены для алгоритма шифрования S-AES

x	Y			
	00	**01**	**10**	**11**
00	9	e	5	1
01	8	b	d	a
10	6	7	f	3
11	c	4	0	2

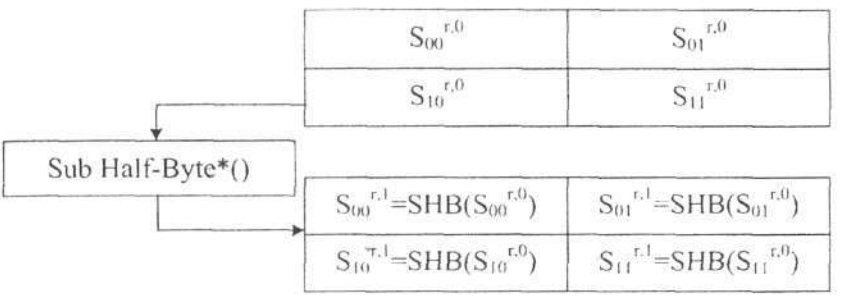


Рис. 58. Применение операции Sub Half-Bytes*()

На рис. 58 и последующих рисунках, описывающих раундовые операции алгоритма шифрования S_AES, нижние два индекса состояния S обозначают его местоположение в таблице состояний, а верхние два индекса обозначают номер раунда, в котором происходит преобразование, и порядковый номер операции в данном раунде соответственно. Так, например, запись $S_{10}^{r,1}$ обозначает блок данных, находящийся на пересечении второй строки и первого столбца раунда r после применения первой раундовой операции **SubHalf-Bytes*()**, а запись $S_{10}^{r,0}$ обозначает входной блок данных раунда r, находящийся в той же позиции, что и предыдущий блок данных.

Рассмотрим более детально, как происходит преобразование с помощью операции **SubHalf-Bytes*()**. Пусть нам надо преобразовать полубайт {a}. Первым делом, как уже отмечалось выше, необходимо найти мультипликативный обратный

ему элемент в поле $GF(2^4)$. Полубайт {a} можно представить в виде многочлена $(x^3 \oplus x)$, а значит, нам необходимо найти такой многочлен A, что

$$(x^3 \oplus x) \bullet A = 1 \bmod (x^4 \oplus x \oplus 1).$$

Поделив неприводимый многочлен $(x^4 \oplus x \oplus 1)$ на многочлен $(x^3 \oplus x)$, легко можно определить, что целой частью от деления будет значение x, а в остатке останется многочлен $(x^2 \oplus x \oplus 1)$. То есть

$$(x^4 \oplus x \oplus 1) \oplus x \bullet (x^3 \oplus x) = x^2 \oplus x \oplus 1. \qquad (6)$$

В свою очередь, поделив многочлен $(x^3 \oplus x)$ на многочлен $(x^2 \oplus x \oplus 1)$, можно определить, что многочлен $(x^2 \oplus x \oplus 1)$ входит в многочлен $(x^3 \oplus x)$ $(x \oplus 1)$ раз, и при этом в остатке остается многочлен $(x \oplus 1)$. Следовательно,

$$(x^3 \oplus x) \oplus ((x \oplus 1) \bullet (x^2 \oplus x \oplus 1)) = x \oplus 1. \qquad (7)$$

Подставив (6) в (7), получим:

$$(x^3 \oplus x) \oplus ((x \oplus 1) \bullet ((x^4 \oplus x \oplus 1) \oplus x \bullet (x^3 \oplus x))) = x \oplus 1;$$
$$(x^3 \oplus x) \oplus ((x \oplus 1) \bullet (x^4 \oplus x \oplus 1)) \oplus ((x \oplus 1) \bullet x \bullet (x^3 \oplus x)) = x \oplus 1;$$
$$((x \oplus 1) \bullet (x^4 \oplus x \oplus 1)) \oplus ((x^2 \oplus x \oplus 1) \bullet (x^3 \oplus x)) = x \oplus 1. \qquad (8)$$

Поделив многочлен $(x^2 \oplus x \oplus 1)$ на многочлен $(x \oplus 1)$, можно определить, что многочлен $(x \oplus 1)$ входит в многочлен $(x^2 \oplus x \oplus 1)$ (x) раз, и при этом в остатке остается единица, т. е.:

$$(x^2 \oplus x \oplus 1) \oplus ((x \oplus 1) \bullet x) = 1. \qquad (9)$$

Подставив (6) и (8) в (9), получим:

$$((x^4 \oplus x \oplus 1) \oplus x \bullet (x^3 \oplus x)) \oplus ((((x \oplus 1) \bullet (x^4 \oplus x \oplus 1)) \oplus$$
$$\oplus ((x^2 \oplus x \oplus 1) \bullet (x^3 \oplus x))) \bullet x) = 1;$$

$$(x^4 \oplus x \oplus 1) \oplus (x \bullet (x^3 \oplus x)) \oplus ((x^2 \oplus x) \bullet (x^4 \oplus x \oplus 1)) \oplus$$
$$\oplus ((x^3 \oplus x^2 \oplus x) \bullet (x^3 \oplus x)) = 1;$$
$$((x^4 \oplus x \oplus 1) \bullet (x^2 \oplus x \oplus 1)) \oplus ((x^3 \oplus x^2) \bullet (x^3 \oplus x)) = 1,$$

или

$$(x^3 \oplus x^2) \bullet (x^3 \oplus x) = 1 \oplus ((x^4 \oplus x \oplus 1) \bullet (x^2 \oplus x \oplus 1)).$$

А значит, искомым многочленом А будет являться многочлен $(x^3 \oplus x^2)$ или значение {c} в шестнадцатеричном виде, или значение {1100} в двоичном.

Вторым этапом является преобразование следующего вида:

$$b_0' = b_0 \oplus b_2 \oplus b_3 \oplus 1 = 0 \oplus 1 \oplus 1 \oplus 1 = 1;$$
$$b_1' = b_1 \oplus b_3 \oplus b_0 \oplus 0 = 0 \oplus 1 \oplus 0 \oplus 0 = 1;$$
$$b_2' = b_2 \oplus b_0 \oplus b_1 \oplus 0 = 1 \oplus 0 \oplus 0 \oplus 0 = 1;$$
$$b_3' = b_3 \oplus b_1 \oplus b_2 \oplus 1 = 1 \oplus 0 \oplus 1 \oplus 1 = 1.$$

Результатом проведенного преобразования является значение {1111} или {f}. По табл. 35 можно убедиться, что на пересечении третьей строки и третьего столбца действительно находится значение {f}.

Операция сдвига строк Shift Rows*()

Эта операция осуществляется аналогично одноименной операции в алгоритме шифрования AES, т.е. первая строка массива данных остается без изменений, а вторая циклически сдвигается влево на один полубайт. На рис. 59 показано применение операции **Shift Rows*()** к шифруемым данным.

Операция перемешивания столбцов Mix Columns*()

При выполнении этой операции столбцы данных рассматриваются как многочлены над $GF(2^4)$ и умножаются по модулю $x^2 \oplus 1$ на многочлен g(x), выглядящий следующим образом:

$$g(x) = \{2\}x \oplus \{3\}.$$

Это может быть представлено в матричном виде следующим образом:

$$\begin{bmatrix} S_{0c}' \\ S_{1c}' \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} S_{0c} \\ S_{1c} \end{bmatrix}, 0 \le c \le 1,$$

где c – номер столбца массива данных. В результате такого умножения полубайты столбца S_{0c} и S_{1c} заменяются соответственно на полубайты:

$$S_{0c}' = (\{3\} \bullet S_{0c}) \oplus (\{2\} \bullet S_{1c}),$$
$$S_{1c}' = (\{2\} \bullet S_{0c}) \oplus (\{3\} \bullet S_{1c}).$$

На рис. 60 показано применение операции **Mix Columns***() к данным, преобразуемым в ходе зашифрования.

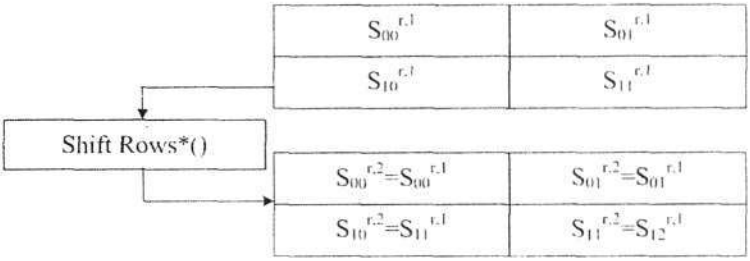


Рис. 59. Преобразование данных с помощью операции Shift Rows*()

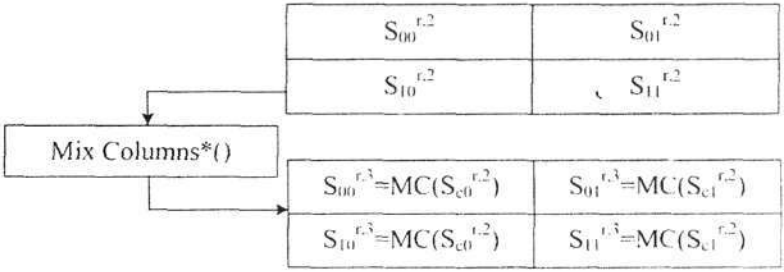


Рис. 60. Применение операции Mix Columns*() к преобразуемым данным

Рассмотрим операцию **Mix Columns*()** более подробно. Пусть нам надо преобразовать столбец:

$S_{00} = \{c\}$
$S_{10} = \{9\}$

Первым делом необходимо представить полубайты S_{00} и S_{10} в виде многочленов в поле $GF(2^4)$. Многочлен $(x^3 \oplus x^2)$ будет эквивалентен байту $S_{00} = \{c\}$, а многочлен $(x^3 \oplus 1)$ – байту $S_{10} = \{9\}$.

Для нахождения значения S_{00}' необходимо выполнить следующие действия:

1. Умножить многочлен $(x^3 \oplus x^2)$ на полубайт {3} или, иначе говоря, на $(x \oplus 1)$ по модулю $(x^4 \oplus x \oplus 1)$:

$$((x^3 \oplus x^2) \bullet (x \oplus 1)) \bmod (x^4 \oplus x \oplus 1) = (x^4 \oplus x^3 \oplus x^2) \bmod (x^4 \oplus x \oplus 1) =$$
$$= (x^4 \oplus x^2) \bmod (x^4 \oplus x \oplus 1) = x^2 \oplus x \oplus 1.$$

2. Умножить многочлен (x3⊕1) на полубайт {2} или, иначе говоря, на x по модулю (x4⊕x⊕1):

$$((x^3 \oplus 1) \bullet x) \bmod (x^4 \oplus x \oplus 1) = (x^4 \oplus x) \bmod (x^4 \oplus x \oplus 1) = 1.$$

3. Сложить по модулю 2 многочлены, полученные в пп. 1 и 2:

$$(x^2 \oplus x \oplus 1) \oplus 1 = x^2 \oplus x.$$

Искомым значением S_{00}' будет являться многочлен $x^2 \oplus x$ или полубайт {6}.

Теперь найдем значение S_{10}'. Для этого:

1. Умножим многочлен $(x^3 \oplus x^2)$ на полубайт {2} или, иначе говоря, на x по модулю $(x^4 \oplus x \oplus 1)$:

$$((x^3 \oplus x^2) \bullet x) \bmod (x^4 \oplus x \oplus 1) = (x^4 \oplus x^3) \bmod (x^4 \oplus x \oplus 1) = x^3 \oplus x \oplus 1.$$

2. Умножим многочлен (x3⊕1) на полубайт {3} или, иначе говоря, на (x⊕1) по модулю (x4⊕x⊕1):

$$((x^3 \oplus 1) \bullet (x \oplus 1)) \bmod (x^4 \oplus x \oplus 1) =$$
$$= (x^4 \oplus x \oplus x^3 \oplus 1) \bmod (x^4 \oplus x \oplus 1) = x^3.$$

3. Сложим по модулю 2 многочлены, полученные в пп. 1 и 2:

$$(x^3 \oplus x \oplus 1) \oplus x^3 = x \oplus 1.$$

Таким образом, искомым значением S_{10}' будет являться многочлен x⊕1 или полубайт {3}.

На рис. 61 наглядно показано преобразование исходного столбца S_{i0} в столбец S_{i0}' с помощью операции **Mix Columns*()**.

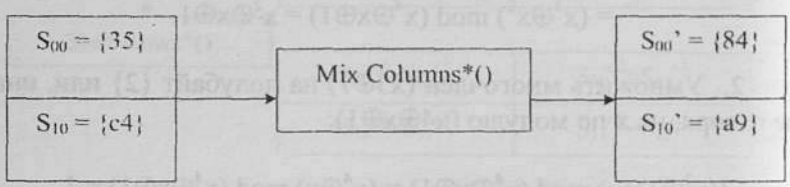


Рис. 61. Преобразование столбца с помощью операции Mix Columns*()

Сложение с раундовым подключом AddRoundKey*

В операции добавления подключа (**AddRoundKey***) подключ добавляется к данным с помощью операции побитового сложения по модулю 2 (операция XOR). На рис. 62 показана операция сложения данных с раундовым подключом.

Обратите внимание, что на рис. 62 исходные данные, складываемые с ключом, имеют верхний индекс in, который означает, что эти данные являются исходными данными, подлежащими зашифрованию с помощью алгоритма шифрования S_AES. Верхний индекс при обозначении полубайтов ключа определяет номер используемого подключа.

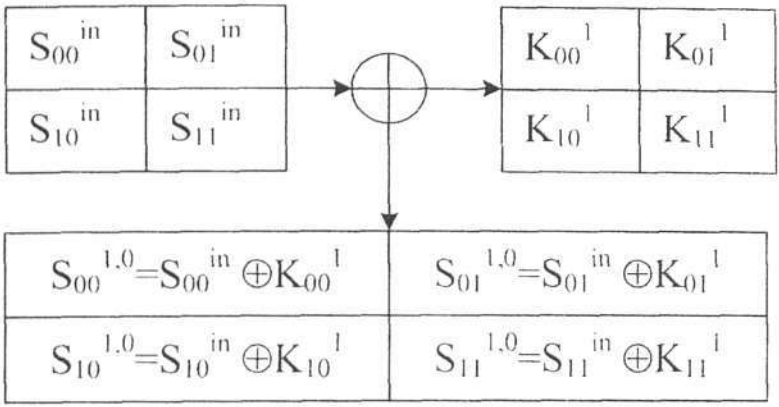


Рис. 62. Операция AddRoundKey*() сложения данных с раундовым подключом

На рис. 63 показан полный алгоритм S_AES. При расшифровании данных вместо операции Sub Half-Bytes*() выполняется инверсное ей преобразование. Неизменным остается преобразование Shift Rows*(). Дело в том, что массив данных в алгоритме шифрования S_AES содержит всего две строки и два столбца. При зашифровании с помощью операции Shift Rows*() происходит циклический сдвиг влево на один полубайт. А значит, при расшифровании необходимо будет провести циклический сдвиг вправо также на один полубайт. Но так как в строке содержится всего два элемента, то, в принципе, все равно в какую сторону их циклически сдвигать. Поэтому при расшифровании смело можно пользоваться операцией Shift Rows*(). Кроме того, все операции используются в обратном порядке так же как и раундовые подключи, т. е. сначала K^3, потом K^2 и последний – K^1. На рис. 64 приведен алгоритм расшифрования данных с помощью алгоритма S_AES.

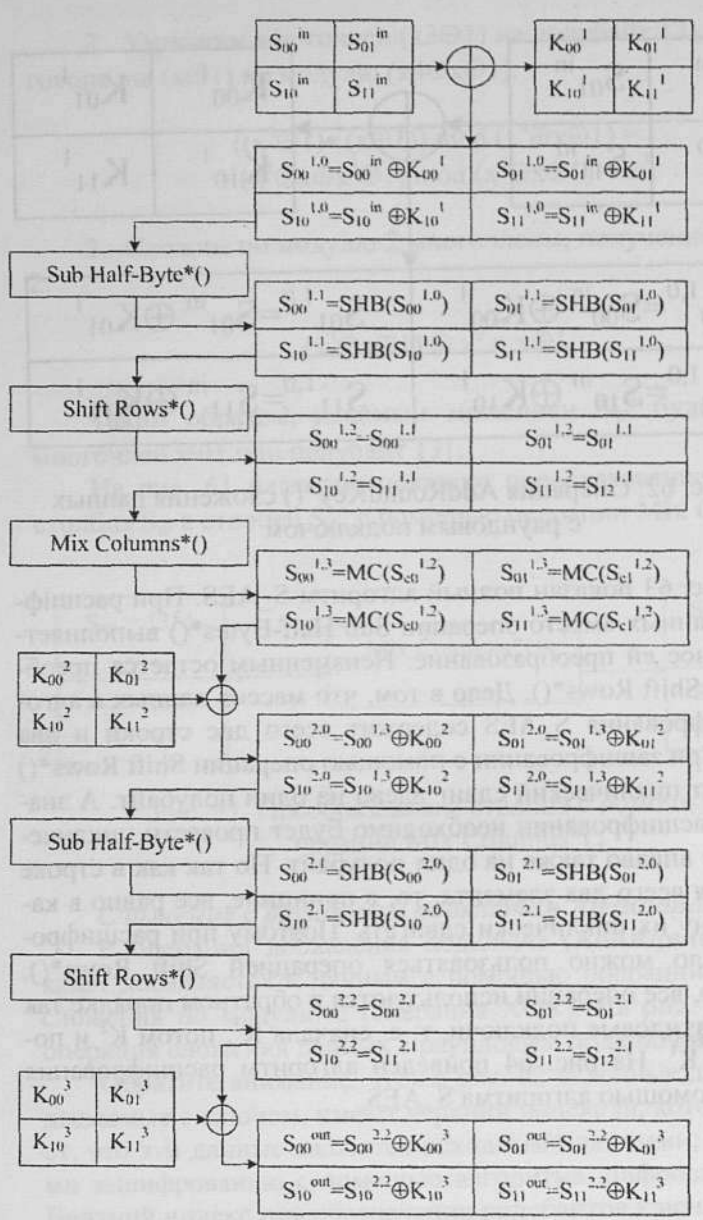


Рис. 63. Алгоритм шифрования S_AES

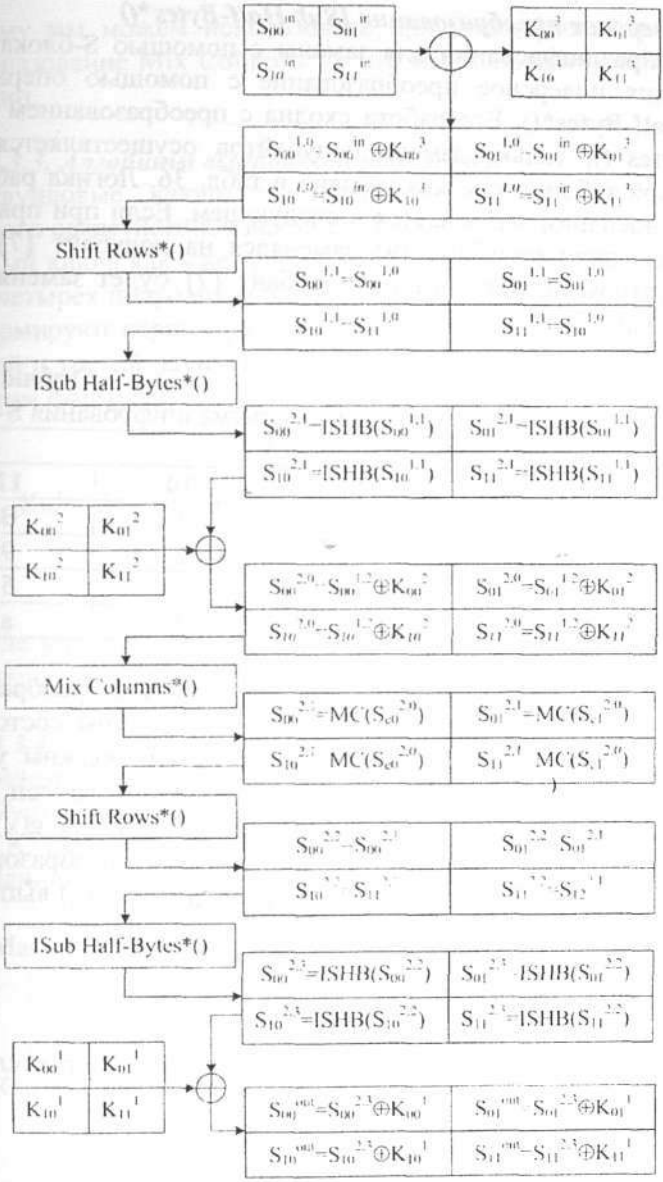


Рис. 64. Алгоритм расшифрования S_AES

Инверсное преобразование ISub Half-Bytes*()

При расшифровании для замены с помощью S-блока используется инверсное преобразование с помощью операции **ISub Half-Bytes*()**. Его работа сходна с преобразованием **Sub Half-Bytes*()**, только замена полубайтов осуществляется по инверсной таблице так, как указано в табл. 36. Логика работы данной операции заключается в следующем. Если при прямом преобразовании полубайт {9} заменялся на полубайт {7}, то при обратном преобразовании полубайт {7} будет заменяться на полубайт {9}.

Таблица 36

Инверсный S-блок замены для алгоритма шифрования S-AES

x	Y			
	00	**01**	**10**	**11**
00	e	3	f	B
01	d	2	8	9
10	4	0	7	5
11	c	6	1	a

При расшифровании неизменным остается и преобразование Mix Columns*(). При расшифровании столбцы состояния рассматриваются как многочлен над $GF(2^4)$ и должны умножаться по модулю $x^2 \oplus 1$ на многочлен $g^{-1}(x)$. Многочлен $g^{-1}(x)$ является мультипликативным обратным многочлена g(x), используемого в операции прямого преобразования MixColumns*(). То есть, для многочленов g(x) и $g^{-1}(x)$ выполняется соотношение

$$g(x) \bullet g^{-1}(x) = \{01\}.$$

Это соотношение будет выполняться только в том случае, если

$$g^{-1}(x) = g(x) = \{2\}x \oplus \{3\},$$

поэтому мы можем использовать при расшифровании то же преобразование Mix Columns*(), что использовали и при шифровании.

4.3.3. Алгоритм выработки подключей

Раундовые ключи вырабатываются из исходного 16-битового секретного ключа по следующей схеме. Исходный 16-битовый ключ, как уже говорилось ранее, можно представить в виде четырех полубайтов K_{00}, K_{10}, K_{01}, K_{11}. Эти четыре полубайта формируют первый раундовый подключ: K_{00}^{1}, K_{10}^{1}, K_{01}^{1}, K_{11}^{1}. Второй и третий раундовые подключи можно получить по следующим формулам:

$$\begin{aligned} K_{00}^{r} &= \text{Sub Half-Bytes*}(K_{11}^{r-1}) \oplus K_{00}^{r-1}; \\ K_{10}^{r} &= \text{Sub Half-Bytes*}(K_{01}^{r-1}) \oplus K_{10}^{r-1} \oplus 2^{r-2}; \\ K_{01}^{r} &= K_{00}^{r} \oplus K_{01}^{r-1}; \\ K_{11}^{r} &= K_{10}^{r} \oplus K_{11}^{r-1}, \end{aligned} \qquad (10)$$

где верхний индекс r – номер вырабатываемого подключа, а преобразование Sub Half-Bytes*() осуществляется по табл. 35.

4.3.4. Пример преобразования данных с помощью алгоритма S_AES

Пусть даны входной блок данных: {7, e, 3, b} и секретный ключ K: {3, e, f, a}.

Запишем эти данные в виде двумерных массивов:

Массив исходных данных:

7	3
e	b

Ключ:

3	f
e	a

Для начала необходимо получить раундовые подключи. Первым раундовым подключом будет являться сам секретный ключ, т. е.

$$K_{00}^{1} = 3; K_{10}^{1} = e; K_{01}^{1} = fK_{11}^{1} = a.$$

Воспользовавшись формулами (10), найдем:

$$K_{00}^{2} = \text{Sub Half-Bytes}^{*}(K_{11}^{1}) \oplus K_{00}^{1} =$$
$$= \text{Sub Half-Bytes}^{*}(a) \oplus 3 = 1 \oplus 3 = c;$$
$$K_{10}^{2} = \text{Sub Half-Bytes}^{*}(K_{01}^{1}) \oplus K_{10}^{1} \oplus 2^{0} =$$
$$= \text{Sub Half-Bytes}^{*}(f) \oplus e \oplus 1 = = 2 \oplus e \oplus 1 = d;$$
$$K_{01}^{2} = K_{00}^{2} \oplus K_{01}^{1} = c \oplus f = 3;$$
$$K_{11}^{2} = K_{10}^{2} \oplus K_{11}^{1} = d \oplus a = 7;$$
$$K_{00}^{3} = \text{Sub Half-Bytes}^{*}(K_{11}^{2}) \oplus K_{00}^{2} =$$
$$= \text{Sub Half-Bytes}^{*}(7) \oplus c = a \oplus c = 6;$$
$$K_{10}^{3} = \text{Sub Half-Bytes}^{*}(K_{01}^{2}) \oplus K_{10}^{2} \oplus 2^{1} =$$
$$= \text{Sub Half-Bytes}^{*}(3) \oplus d \oplus 2 = 1 \oplus d \oplus 2 = e;$$
$$K_{01}^{3} = K_{00}^{3} \oplus K_{01}^{2} = 6 \oplus 3 = 5;$$
$$K_{11}^{2} = K_{10}^{3} \oplus K_{11}^{2} = e \oplus 7 = 9.$$

Таким образом, получили три раундовых подключа:

Первый подключ К1:

3	f
e	a

Второй подключ К2:

c	3
d	7

Третий подключ К3:

6	5
e	9

При зашифровании с помощью алгоритма S_AES начальной операцией является сложение исходных данных с первым раундовым подключом. В результате получим:

7 ⊕ 3 = 4	3 ⊕f = c
e ⊕ e = 0	b⊕a = 1

Затем произведем замену полубайтов с помощью табл. 35 и получим:

8	c
9	e

Теперь сдвинем вторую строку на один полубайт влево:

8	c
e	9

и произведем перемешивание столбцов с помощью операции MixColumns*(). Для начала представим табличные данные в виде многочленов. Итак, $S_{00} = \{8\} = x^3$, $S_{10} = \{e\} = x^3 \oplus x^2 \oplus x$, $S_{01} = \{c\} = x^3 \oplus x^2$, $S_{11} = \{9\} = x^3 \oplus 1$.

Теперь можно приступить к нахождению новых элементов массива:

$$S_{00}' = (\{3\} \bullet S_{00}) \oplus (\{2\} \bullet S_{10}) = ((x \oplus 1) \bullet x^3) \oplus (x \bullet (x^3 \oplus x^2 \oplus x)) =$$
$$= (x^4 \oplus x^3) \oplus (x^4 \oplus x^3 \oplus x^2) = x^3 \oplus x \oplus 1 \oplus x^3 \oplus x^2 \oplus x \oplus 1 = x^2 = \{4\};$$
$$S_{10}' = (\{2\} \bullet S_{00}) \oplus (\{3\} \bullet S_{10}) = (x \bullet x^3) \oplus ((x \oplus 1) \bullet (x^3 \oplus x^2 \oplus x)) =$$
$$= (x^4) \oplus (x^4 \oplus x^3 \oplus x^2 \oplus x^3 \oplus x^2 \oplus x) = (x^4) \oplus (x^4 \oplus x) = x \oplus 1 \oplus 1 =$$
$$= x = \{2\};$$
$$S_{01}' = (\{3\} \bullet S_{01}) \oplus (\{2\} \bullet S_{11}) = ((x \oplus 1) \bullet (x^3 \oplus x^2)) \oplus (x \bullet (x^3 \oplus 1)) =$$
$$= (x^4 \oplus x^3 \oplus x^3 \oplus x^2) \oplus (x^4 \oplus x) = (x^4 \oplus x^2) \oplus (x^4 \oplus x) = x^2 \oplus x \oplus 1 \oplus 1 =$$
$$= x^2 \oplus x = \{6\};$$
$$S_{11}' = (\{2\} \bullet S_{01}) \oplus (\{3\} \bullet S_{11}) = (x \bullet (x^3 \oplus x^2)) \oplus ((x \oplus 1) \bullet (x^3 \oplus 1)) =$$
$$= (x^4 \oplus x^3) \oplus (x^4 \oplus x \oplus x^3 \oplus 1) = x^3 \oplus x \oplus 1 \oplus x^3 = x \oplus 1 = \{3\}.$$

Получили следующий массив преобразованных данных:

4	6
2	3

Следующим шагом является сложение полученных данных со вторым раундовым подключом, т. е.

4 ⊕ c = 8	6 ⊕ 3 = 5
2 ⊕d = f	3 ⊕ 7 = 4

Теперь еще раз произведем замену полубайтов с помощью табл. 35 и получим:

6	b
2	8

Теперь сдвинем вторую строку на один полубайт влево:

6	b
8	2

Заключительной операцией является сложение с третьим раундовым подключом:

6 ⊕ 6 = 0	b⊕ 5 = e
8 ⊕ e = 6	2 ⊕ 9 = b

Итак, в результате шифрования данных с помощью алгоритма шифрования S_AES, мы получили в результате значение {0, 6, e, b}.

5. РЕЖИМЫ ШИФРОВАНИЯ ДЛЯ СИММЕТРИЧНЫХ БЛОЧНЫХ ШИФРОВ

Существует довольно большое число различных способов объединять симметричные блочные шифры для получения нового, более стойкого варианта шифра. Изначально большинство схем предназначалось для использования их совместно с алгоритмом DES, который длительное время был стандартом США и являлся базовым алгоритмом. Однако в большинстве своем разработанные схемы оказывались универсальными и в дальнейшем могли применяться для шифрования данных с использованием других блочных шифров. Всего выделяют четыре основных режима работы блочных шифров (рис. 65). В данном разделе мы рассмотрим как индивидуальные режимы шифрования, предназначенные для конкретных алгоритмов шифрования (например, DES или ГОСТ), так и универсальные, рассчитанные на применение большого круга блочных шифров.

Рис. 65. Основные режимы работы блочных шифров

5.1. Основные режимы работы блочных шифров

5.1.1. Режим электронной кодовой книги (Electronic codebook, ECB)

Режим электронной кодовой книги является самым простым и самым естественным режимом. При его использовании весь объем информации разбивается на блоки (величина блока зависит от используемого алгоритма шифрования). После этого каждый блок зашифровывается на секретном ключе и помеща-

ется в зашифрованный файл. В случае, если последний блок имеет размер меньше, чем требуется для обработки алгоритмом шифрования, производится дополнение блока. Обычно все оставшееся пространство за исключением последнего байта заполняется нулями, а в последний байт заносится размер истинного блока данных или размер дополненной информации. Так как блоки обрабатываются независимо друг от друга, то производить их обработку можно параллельно, тем самым ускоряя процесс шифрования. Схема зашифрования/дешифрования данных в режиме ECB представлена на рис. 66. В формульном виде описать работу режима можно следующим образом:

при зашифровании $C_i = E(P_i, K)$;
при дешифровании $P_i = D(C_i, K)$.

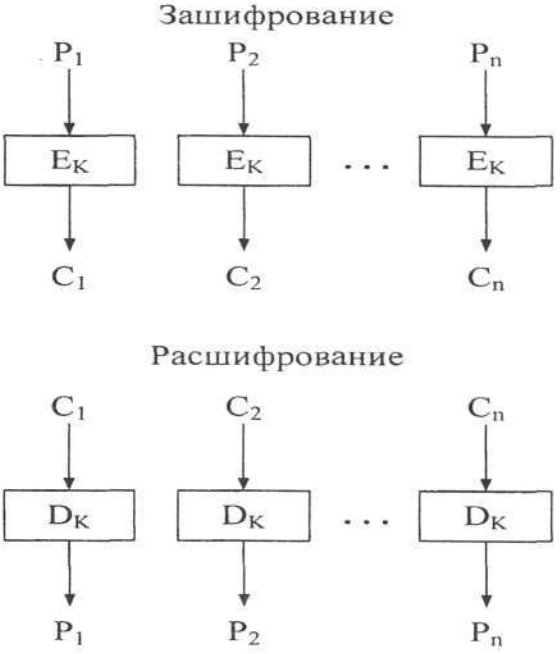


Рис. 66. Схема работы режима ECB

Это интересно

Использование режима ECB не всегда является безопасным. В своей книге [2] Б. Шнайер приводит интересный пример того, как злоумышленник может использовать слабые стороны режима ECB так, чтобы, не зная ключа или даже алгоритма, иметь возможность обмануть предполагаемого получателя.

Для иллюстрации этой проблемы в книге [2] приводится пример системы передачи денег, которая переводит деньги из банка в банк. Допустим, используется какой-то стандартный формат перевода денег между банками. Например, сообщение, пересылаемое банками друг другу, занимает ровно 12 блоков, каждый блок обрабатывается в режиме ECB. При этом блоки распределяется следующим образом:

Банк 1: Передача 1.5 блока
Банк 2: Прием 1.5 блока
Имя вкладчика: 6 блоков
Счет вкладчика: 2 блока
Сумма вклада: 1 блок

Злоумышленник, который прослушивает линию связи между банками, может использовать эту информацию для того, чтобы пополнить свой банковский счет. Сначала, он записывает все шифрованные сообщения из Банка 1 в Банк 2. Затем, он переводит $100 из Банка 1 в Банк 2. Позже он повторяет эту операцию еще раз. С После этого он проверяет все ранее перехваченные и записанные сообщения, находит пару идентичных сообщений. Это как раз и будут те сообщения, с помощью которых он дважды перевел $100 на свой счет. Если он находит несколько пар одинаковых сообщений (что больше похоже на реальную жизнь), он делает еще один денежный перевод и записывает результат. В конце концов, он сможет выделить сообщение, которым был проведен именно его перевод.

Как только он выявил свое сообщение, он может отправить это сообщение по каналу связи в Банк 2 когда захочет, тем самым каждый раз зачисляя на свой счет $100.

На первый взгляд банки могут легко пресечь это, добавляя метки времени к своим сообщениям.

Метка даты/времени: 1 блок
Банк 1: Передача 1.5 блока
Банк 2: Прием 1.5 блока
Имя вкладчика: 6 блоков
Счет вкладчика: 2 блока
Сумма вклада: 1 блок

В такой системе два идентичных сообщения будут легко обнаружены. Тем не менее с помощью метода, называемого повтором блока, злоумышленник все же сможет обогатиться. На рис. 67 показано, что злоумышленник может собрать восемь блоков шифр-текста, соответствующих его имени и номеру счета: блоки с 5 по 12.

1	2	3	4	5	6	7	8	9	10	11	12
Метка времени	Банк-получатель		Банк-отправитель	Имя вкладчика					Счет вкладчика		Сумма

Рис. 67. Блоки шифрования в записи приведенного примера

Он перехватывает сообщения из Банка 1 в Банк 2 и заменяет блоки с 5 по 12 сообщения байтами, соответствующими его имени и номеру счета. Затем он посылает измененные сообщения в Банк 2. В этом случае ему не нужно знать, кто был отправителем денег, ему даже не нужно знать переводимую сумму (хотя он может связать подправленное сообщение с соответствующим увеличением своего счета и определить блоки, соответствующие определенным денежным суммам). Он просто изменяет имя и номер счета на свои собственные и следит за ростом своих доходов.

5.1.2. Режим сцепления блоков (cipher block chaining, CBC)

Для борьбы с возможными злоумышленниками и предотвращения возможностей подлога, описанных в

подразд. 5.1.1, используют схемы, обеспечивающие связь шифруемых блоков друг с другом. Самым простым примером такой связи является сцепление, которое добавляет к блочному алгоритму шифрования механизм обратной связи: результаты шифрования предыдущих блоков влияют на шифрование текущего блока. В результате получается, что каждый блок шифртекста зависит не только от шифруемого блока открытого текста, но и от всех предыдущих блоков открытого текста.

В режиме сцепления блоков шифра (cipher block chaining, CBC) перед шифрованием данных выполняется операция сложения по модулю 2 блока открытого текста и блока предыдущего шифрованного текста. Для обработки первого блока используется операция сложения блока открытого текста с вектором инициализации. На рис. 68 приведена схема работы режима CBC для шифрования и расшифрования данных. Расшифрирование является обратной операцией (рис. 68). Сначала блок шифр-текста расшифровывается как обычно. Затем следующий блок расшифровывается и складывается по модулю 2 с предыдущим расшифрованным блоком. Для расшифрования первого блока сообщения, расшифрованный блок складывается с тем же вектором инициализации, который использовался для зашифрования данных.

Математически это можно выразить следующим образом:

$$Ci = Ek(Pi \oplus Ci - 1),$$
$$Pi = Ci{-}1 \oplus Dk(Ci).$$

Для того чтобы одинаковые блоки сообщения шифровались по-разному, используется вектор инициализации IV. Он представляет собой блок данных, длина которого равна длине блока сообщения, обрабатываемого с помощью используемого симметричного шифра. Первый текст сообщения складывается по модулю 2 с вектором инициализации. Таким образом, два одинаковых сообщения при использовании различных значений IV будут зашифрованы по-разному. Это особенно актуально для тех сообщений, которые имеют одинаковую структуру. Например, писем, заголовок которых начинается со слов «От ко-

го». При этом вектор инициализации IV не должен храниться в секрете, он может передаваться открыто вместе с шифр-текстом.

В случае, когда последний блок шифрования оказывается меньше, чем нужно для обработки, применяется дополнение блоков. Оно используется так же, как и в режиме ECB.

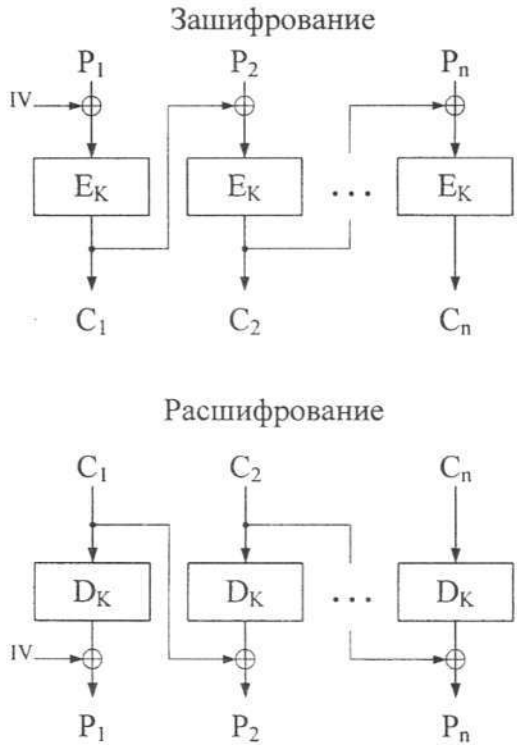


Рис. 68. Схема работы режима CBC

5.1.3. Режим обратной связи по выходу (Output-Feedback, OFB)

В двух предыдущих режимах шифрование осуществялось поблочно. То есть минимальный объем зашифрованной информации был равен одному блоку (или одному дополненному

блоку). Два следующих режима могут за один раз зашифровывать порцию информации, меньшую чем один блок. Это режимы обратной связи по шифру и обратной связи по выходу. В частности, это может быть полезно, когда вам необходимо зашифровать небольшую порцию информации, например, пароль или ключ. В настоящем подразделе мы рассмотрим принцип работы режима обратной связи по выходу, схема которого представлена на рис. 69.

Рассмотрим как работает режим обратной связи по выходу. Само шифрование сообщения P осуществялется путем его сложения с некоторой псевдослучайной последовательностью (ПСП), для выработки которой и используется симметричный блочный шифр. Так как операция сложения по модулю 2 обратна сама себе, то для расшифрования сообщения необходимо выработать ПСП точно таким же образом и сложить результат с зашифрованным сообщением.

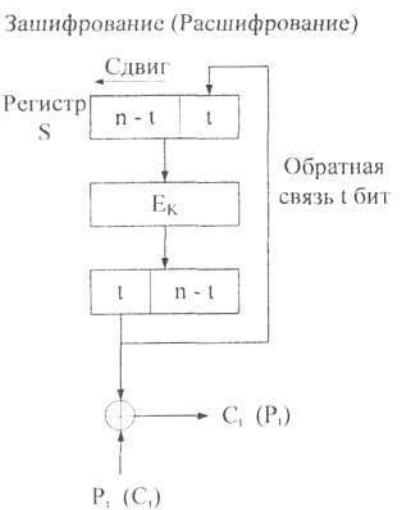


Рис. 69. Схема работы режима OFB

Для выработки ПСП берется содержимое регистра S. Первоначально в него заносится значение вектора инициализации

(требования, которые предъявляются к вектору инициализации, были рассмотрены в подразд. 5.1.2 и распространяются на все рассматриваемые режимы шифрования). Содержимое регистра S зашифровывается на ключе K с помощью симметричного блочного шифра. Из полученного шифр-текста $E_K(S)$ берутся первые (старшие) t разрядов, которые представляют собой первую ПСП и используются для зашифрования первого сообщения. Для выработки второй ПСП в регистре S производится сдвиг влево (в сторону старших разрядов) на t позиций. В освободившиеся разряды заносится первая ПСП (t бит, которые использовались для зашифрования первого блока). Далее действия повторяются. Содержимое регистра S зашифровывается на ключе K, берутся первые (старшие) t бит шифра, складываются со вторым текстом по модулю 2, образуя второй шифр. Выполняется обратная связь, путем занесения t разрядов ПСП в младшие разряды регистра S и т.д.

Обозначим через T(x) функцию, которая берет от значения x только первые (старшие) t бит. Тогда режим шифрования OFB можно выразить математически следующим образом:

$$Ci = Pi \oplus T(Ek(S)),$$
$$Pi = Ci \oplus T(Ek(S)),$$
$$S0 = IV,\ Si = (S(i-1) << t) \| T(Ek(S)).$$

5.1.4. Режим обратной связи по шифру (Cipher-Feedback, CFB)

Режим шифрования обратной связи по шифру очень схож с работой режима обратной связи по выходу, за тем исключением, что в младшие разряды регистра S будет записываться не значение ПСП, а само значение шифр-текста. Схема работы режима OFB показана на рис. 70.

Как и в предыдущем режиме, шифрование открытого текста будет осуществляться путем сложения его с ПСП. Разница будет заключаться в выработке этой ПСП, а именно: в том, какие данные заносятся в регистр S. Соответственно расшифрование будет заключаться в сложении зашифрованных данных с той же ПСП, что была использована при зашифровании.

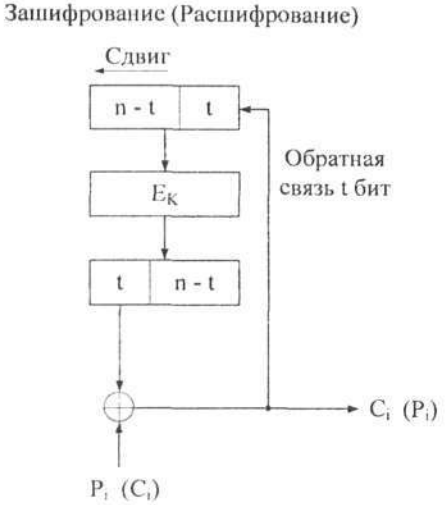


Рис. 70. Схема работы режима CFB

Если обозначить через T(x) функцию, которая берет от значения x только первые (старшие) t бит, то режим шифрования CFB можно выразить математически следующим образом:

$$Ci = Pi \oplus T(Ek(S)),$$
$$Pi = Ci \oplus T(Ek(S)).$$
$$S0 = IV,\ Si = (S(i-1) << t) \| Ci.$$

5.2. Специальные режимы работы алгоритма DES

5.2.1. Двойной DES

Появление данного режима шифрования было продиктовано желанием повысить криптостойкость алгоритма шифрования DES за счет увеличения объема секретной информации (т. е. ключа). Идея заключалась в том, чтобы зашифровать данные на одном секретном ключе, а потом результат зашифровать второй раз на втором секретном ключе. В результате ключевая информация увеличивалась вдвое, т. е. вместо 56 бит использовалось 112. Общий вид режима шифрования «Двойной DES» приведен на рис. 71.

На вход схемы шифрования поступает сообщение P, которое под воздействием ключа K1 зашифровывается и образуется значение X. Значение X в свою очередь под воздействием ключа K2 зашифровывается и образуется значение C. В формульном виде это можно выразить следующим образом:

$$C = E(E(P,K1), K2).$$

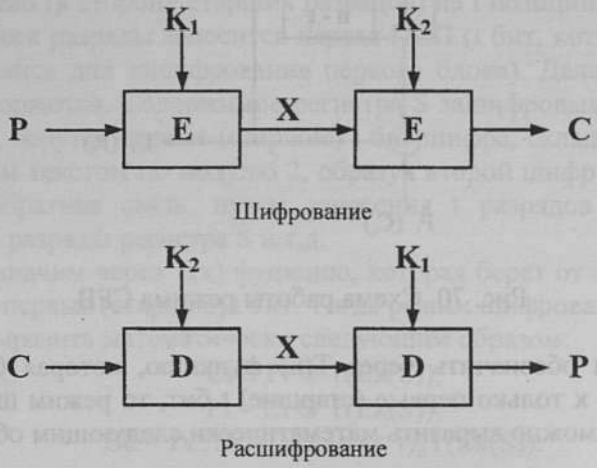


Рис. 71. Двойное шифрование DES

При расшифровании ключи применяются в обратном порядке и используется процедура расшифрования, обратная процедуре зашифрования. Сначала шифр-текст C расшифровывается на ключе K2, тем самым преобразуясь в сообщение X. А потом сообщение X расшифровывается на ключе K1 и преобразуется в сообщение P:

$$P = D(D(C, K2), K1).$$

Казалось бы, что при использовании в качестве базового шифра алгоритма DES, итоговая длина ключа составляет 56x2 = 112 бит, что должно значительно увеличивать криптоаналити-

ческую стойкость шифра. Однако вскоре было показано, что при использовании метода атаки «Встреча посередине» можно осуществить поиск секретных ключей, сделав всего 2^{57} операций зашифрования/расшифрования. Таким образом, оказалось, что вместо ожидаемой стойкости перебора 2^{112} вариантов, фактическая стойкость свелась к перебору 2^{57}. При этом скорость обработки информации снизилась вдвое, так как каждый блок информации необходимо обработать дважды.

Рассмотрим подробнее, как работает метод анализа «Встреча посередине». С одной стороны, выполняют зашифрование исходного значения открытого текста P на всех возможных значениях для ключа K1 (всего таких возможных значений 2^{56}) и получают возможные значения X. С другой стороны, выполняют расшифрование шифр-текста C на всех возможных значениях для ключа K2 (всего таких возможных значений 2^{56}) и тоже получают возможные значения X. После этого просматривают обе таблицы значений. Те строки, в которых значения X совпадут, дадут информацию о значениях секретного ключа (рис. 72). Таким образом, необходимо построить две таблицы, перебрав дважды 2^{56} значений, что в итоге сводится к перебору 2^{57}.

5.2.2 Тройной DES с двумя ключами

Двойной DES не оправдал надежд разработчиков. Поэтому для устранения проблемы, возникшей при применении метода атаки «Встреча посередине», первоначально решено было использовать тройное шифрование с использованием трех разных ключей. Это должно было увеличить сложность задачи криптоанализа. Но при этом существенным недостатком такого подхода является использование большого ключа 56x3 = 168 битов, который во времена DES считался слишком громоздким (хотя в настоящий момент используются секретные ключи до 256 бит, 20 лет назад хранить и обмениваться таким объемом информации было проблематично).

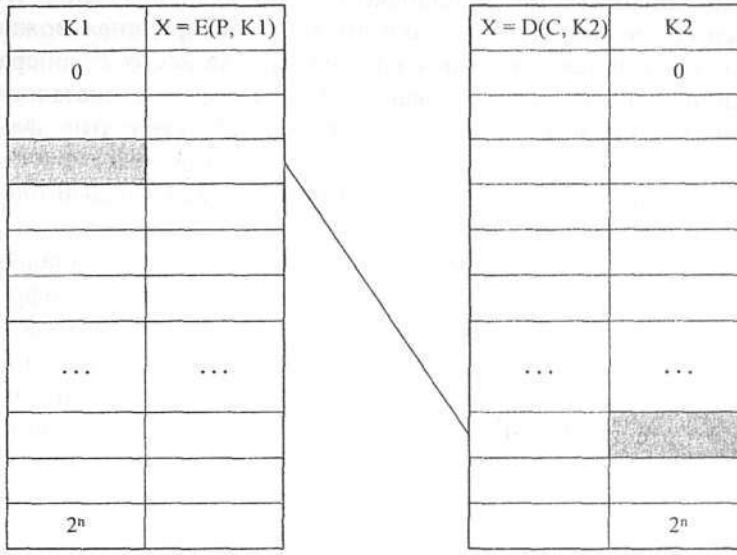


Рис. 72. Анализ режима «Двойной DES» с использованием метода «Встреча посередине»

В качестве решения данной проблемы Уолтер Тачман (Walter Tuchman) предложил к использованию схему тройного шифрования с использованием только двух ключей (рис. 73). В соответствии с предложенной схемой на вход алгоритма шифрования поступает сообщение P, которое под воздействием ключа K1 зашифровывается и образуется значение A. Значение A, в свою очередь, под воздействием ключа K2 расшифровывается и образуется значение B. В заключении значение B зашифровывается на ключе K1, в результате чего образуется шифртекст C. В формульном виде это можно выразить следующим образом:

$$C = E(D(E(P,K1), K2), K1).$$

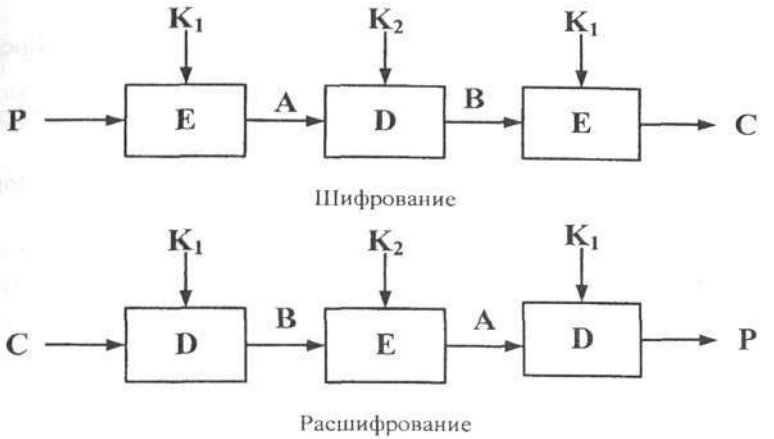


Рис. 73. Тройной DES с двумя ключами

При расшифровании используются инверсные операции: там, где использовалось зашифрование, используется расшифрование, и наоборот. Сначала шифр-текст С расшифровывается на ключе К1, тем самым преобразуясь в сообщение В. Сообщение В зашифровывается на ключе К2 и преобразуется к значению А. В заключение сообщение А расшифровывается на ключе К1 и преобразуется в исходное сообщение Р:

$$P = D(E(D(C, K1), K2), K1).$$

5.3. Специальные режимы для нового стандарта шифрования ГОСТ

В настоящий момент рассматривается проект стандарта для использования различных режимов шифрования с использованием симметричных блочных шифров. В данном проекте стандарта определены следующие режимы работы алгоритмов блочного шифрования:

- режим простой замены (Electronic Codebook, ECB);
- режим гаммирования (Counter, CTR-ГОСТ);
- режим гаммирования с обратной связью по выходу (Output Feedback, OFB-ГОСТ);

– режим простой замены с зацеплением (Cipher Block Chaining, CBC-ГОСТ);

– режим гаммирования с обратной связью по шифр-тексту (Cipher Feedback, CFB-ГОСТ);

– режим выработки имитовставки (Message Authentication Code algorithm MAC-ГОСТ).

Данные режимы могут использоваться в качестве режимов для блочных шифров с произвольной длиной блока n. Рассмотрим каждый их них более подробно.

5.3.1. Вспомогательные операции

Дополнение сообщения

Отдельные из описанных ниже режимов работы (режим гаммирования, режим гаммирования с обратной связью по выходу, режим гаммирования с обратной связью по шифр-тексту) могут осуществлять криптографическое преобразование сообщений произвольной длины так же, как и рассмотренные выше режимы OFB и CFB. Для других режимов (режим простой замены, режим простой замены с зацеплением) требуется, чтобы длина сообщения была кратна некоторой величине ℓ (т. е. для проектного алгоритма ГОСТ ℓ может быть равно 64 или 128 в соответствии с шифруемым размером блока). В последнем случае при работе с сообщениями произвольной длины необходимо применение процедуры дополнения сообщения до требуемой длины. Возможно использование трех разных процедур дополнения. Обозначим $P \in V^*$ исходное сообщение, подлежащее зашифрованию, как $P \in V^*$. Тогда

Процедура 1. Пусть $|P| \equiv r \bmod \ell$. Положим

$$P^* = \begin{cases} P, \text{если } r = 0, \\ P||0^{l-r}, \text{иначе.} \end{cases}$$

В процедуре 1 в случае, если сообщение не кратно размеру блока, недостающее количество битов заполняется нулями. Описанная процедура в некоторых случаях не обеспечивает однозначного восстановления исходного сообщения. Например, результаты дополнения сообщений P1, такого что

$|P_1| = \ell \cdot q - 1$ для некоторого q, и $P_2 = P_1||0$ будут одинаковы. В этом случае для однозначного восстановления необходимо дополнительно знать длину исходного сообщения.

Процедура 2. Пусть $|P| \equiv r \bmod \ell$. Положим

$$P^* = P||1||0^{\ell-r-1}.$$

В процедуре 2 блок дополняется следующим образом. Сначала записывается бит, равный единице, а все остальное пространство заполняется нулями. Данная процедура обеспечивает однозначное восстановление исходного сообщения. При этом если длина исходного сообщения кратна ℓ, то длина дополненного сообщения будет увеличена.

Процедура 3. Пусть $|P| \equiv r \bmod \ell$. В зависимости от значения r возможны случаи:

- если $r = n$, то последний блок не изменяется: $P^* = P$;
- если $r < n$, то применяется процедура 2.

Данная процедура обязательна для режима выработки имитовставки и не рекомендуется для использования в других режимах. Выбор конкретной процедуры дополнения предоставляется разработчику информационной системы и/или регламентируется другими нормативными документами.

Выработка начального значения

В некоторых режимах работы используются величины, начальное значение которых вычисляется на основании синхропосылки (вектора инициализации) IV. Обозначим через m суммарную длину указанных величин. Будем обозначать процедуру выработки начального значения через I_m: $V_{|IV|} \rightarrow V_m$ и называть процедурой инициализации. Будем называть процедуру инициализации тривиальной, если $I_{|IV|} = IV$. Если не оговорено иное, будем считать, что используется тривиальная процедура инициализации на основе синхропосылки необходимой длины. Во всех описываемых в настоящем проекте стандарта режимах работы не требуется обеспечение конфиденциальности синхропосылки. Вместе с тем процедура выработки синхропосылки должна удовлетворять одному из следующих требований.

Во-первых, значения синхропосылки для режимов простой замены с зацеплением и гаммирования с обратной связью по шифр-тексту необходимо выбирать случайно, равновероятно и независимо друг от друга из множества всех допустимых значений. В этом случае значение каждой используемой синхропосылки IV должно быть непредсказуемым (случайным или псевдослучайным): зная значения всех других используемых синхропосылок, значение IV нельзя определить с вероятностью большей, чем $2^{-|IV|}$.

Во-вторых, все значения синхропосылок, выработанных для зашифрования на одном и том же ключе в режиме гаммирования, должны быть уникальными, т. е. попарно различными. Для выработки значений синхропосылок может быть использован детерминированный счетчик.

В-третьих, значение синхропосылки для режима гаммирования с обратной связью по выходу должно быть либо непредсказуемым (случайным или псевдослучайным), либо уникальным.

Режим простой замены не предусматривает использования синхропосылки.

Процедура усечения

В некоторых режимах используется усечение строк длиной n до строк длиной s, $s \leq n$, с использованием функции $T_s = MSB_s$, т. е. в качестве операции усечения используется операция взятия бит с большими номерами (старших значащих битов).

5.3.2. Режим простой замены (ECB - ГОСТ)

Это самый простой режим шифрования из всех рассматриваемых режимов. Он не используется для шифрования больших объемов информации, однако является основой, можно даже сказать «строительным кирпичиком» для построения пяти оставшихся режимов шифрования. Однако небольшие объемы данных, такие как, например, пароль или ключ, могут быть зашифрованы с помощью режима ECB. Длина сообщений, зашифровываемых в режиме простой замены, должна быть кратна длине блока базового алгоритма блочного шифрования n, поэтому, при необходимости, к исходному сообщению должна

быть предварительно применена процедура дополнения. Зашифрование (расшифрование) в режиме простой замены заключается в зашифровании (расшифровании) каждого блока текста с помощью базового алгоритма блочного шифрования.

Зашифрование. Открытый и, при необходимости, дополненный текст $P \in V^*$, $|P| = n \cdot q$, разбивается на отдельные блоки и представляется в виде: $P = P_1||P_2||\ldots||P_q$, $P_i \in V_n$, $i = 1, 2, \ldots, q$. Блоки шифр-текста вычисляются по следующему правилу:

$$C_i = e_K(P_i), \; i = 1, 2, \ldots, q.$$

Результирующий шифр-текст имеет вид

$$C = C_1||C_2||\ldots||C_q.$$

Зашифрование в режиме простой замены проиллюстрировано на рис. 74.

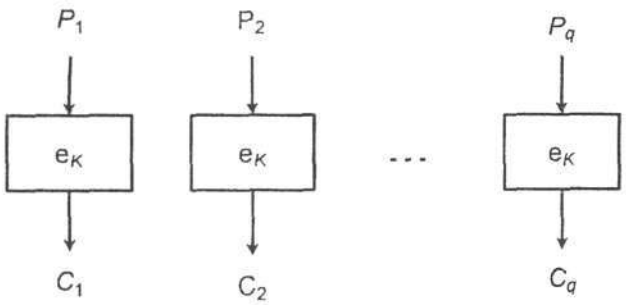


Рис. 74. Зашифрование в режиме простой замены

Расшифрование. Шифр-текст представляется в виде $C = C_1||C_2||\ldots||C_q$, $C_i \in V_n$, $i = 1, 2, \ldots, q$. Блоки открытого текста вычисляются по следующему правилу:

$$P_i = d_K(C_i), \; i = 1, 2, \ldots, q.$$

Исходный (дополненный) открытый текст имеет вид $P = P_1||P2||...||P_q$. Если к исходному открытому тексту была применена процедура дополнения, то после расшифрования следует произвести обратную процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения. Расшифрование в режиме простой замены проиллюстрировано на рис. 75.

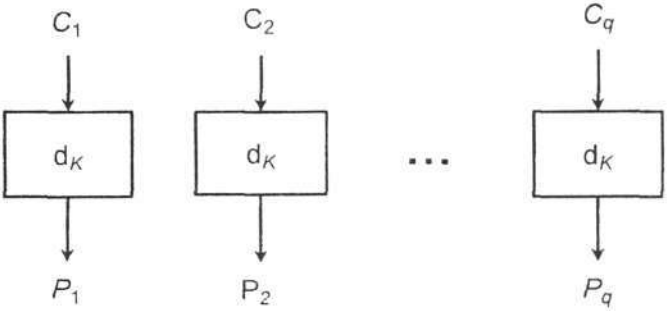


Рис. 75. Расшифрование в режиме простой замены

5.3.3. Режим гаммирования (CTR-ГОСТ)

Режим гаммирования для алгоритма ГОСТ является одним из основных режимов шифрования больших объемов данных. Схема работы данного режима схожа с режимом работы поточных шифров. Сам алгоритм ГОСТ в режиме простой замены используется для выработки псевдослучайной последовательности (ПСП), которая складывается по модулю 2 с блоком открытого текста, тем самым зашифровывая (или расшифровывая) его.

Параметром режима гаммирования является целочисленная величина s, такая, что $0 < s \leq n$. При использовании режима гаммирования не требуется применение процедуры дополнения сообщения. Для зашифрования (расшифрования) каждого отдельного открытого текста на одном ключе используется значение уникальной синхропосылки $IV \in V_{\frac{n}{2}}$. Зашифрование в режиме гаммирования заключается в покомпонентном сложении открытого текста с гаммой шифра, которая вырабатывается

блоками длиной s путем зашифрования последовательности значений счетчика $CTR_i \in V_n$, i = 1, 2, ..., базовым алгоритмом блочного шифрования с последующим усечением. Начальным значением счетчика является $CTR_1 = I_n(IV) = IV \| 0^{\frac{n}{2}}$. Последующие значения счетчика вырабатываются с помощью функции Add: $V_n \to V_n$ следующим образом:

$$CTR_{i+1} = Add(CTR_i) = Vec_n(Int_n(CTR_i) \boxplus_n 1).$$

Зашифрование. Открытый текст $P \in V^*$ разбивается на блоки и представляется в виде $P = P_1 \| P_2 \| ... \| P_q$, $P_i \in V_s$, i = 1, 2, ..., q–1, $P_q \in V_r$, $r \le s$. Блоки шифр-текста вычисляются по следующему правилу:

$$C_i = P_i \oplus T_s(e_K(CTR_i)),\ i = 1, 2, ..., q-1,$$
$$C_q = P_q \oplus T_r(e_K(CTR_q)).$$

Результирующий шифр-текст имеет вид $C = C_1 \| C_2 \| ... \| C_q$. Зашифрование в режиме гаммирования проиллюстрировано на рис. 76.

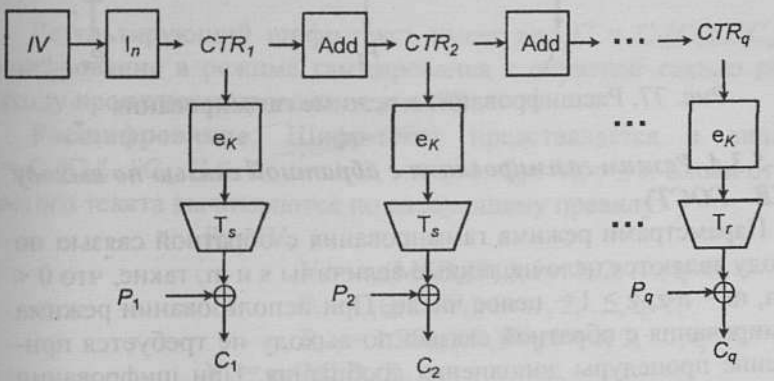


Рис. 76. Зашифрование в режиме гаммирования

Расшифрование. Шифр-текст представляется в виде $C = C_1||C_2||...||C_q$, $C_i \in V_s$, $i = 1, 2, \ldots, q-1$, $C_q \in V_r$, $r \le s$. Блоки открытого текста вычисляются по следующему правилу:

$$P_i = C_i \oplus T_s(e_K(CTR_i)),\ i = 1, 2, \ldots, q-1,$$
$$P_q = C_q \oplus T_r(e_K(CTR_q)).$$

Исходный открытый текст имеет вид $P = P_1||P_2||...||P_q$. Расшифрование в режиме гаммирования проиллюстрировано на рис. 77.

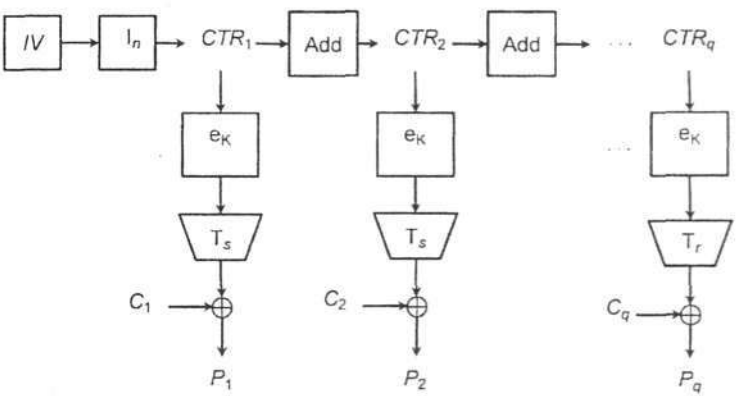


Рис. 77. Расшифрование в режиме гаммирования

5.3.4. Режим гаммирования с обратной связью по выходу (OFB - ГОСТ)

Параметрами режима гаммирования с обратной связью по выходу являются целочисленные величины s и m, такие, что $0 < s \le n$, $m = n \cdot z$, $z \ge 1$ – целое число. При использовании режима гаммирования с обратной связью по выходу не требуется применение процедуры дополнения сообщения. При шифровании на одном ключе для каждого отдельного открытого текста используется значение уникальной или непредсказуемой (случайной или псевдослучайной) синхропосылки $IV \in V_m$. При шифровании в режиме гаммирования с обратной связью по выходу используется двоичный регистр сдвига R длиной m. Начальным

заполнением регистра является значение синхропосылки IV. Зашифрование в режиме гаммирования с обратной связью по выходу заключается в покомпонентном сложении открытого текста с гаммой шифра, которая вырабатывается блоками длиной s. При вычислении очередного блока гаммы выполняется зашифрование n разрядов регистра сдвига с большими номерами базовым алгоритмом блочного шифрования. Затем заполнение регистра сдвигается на n бит в сторону разрядов с большими номерами, при этом в разряды с меньшими номерами записывается полученный выход базового алгоритма блочного шифрования. Блок гаммы вычисляется путем усечения выхода базового алгоритма блочного шифрования.

Зашифрование. Открытый текст $P \in V^*$ представляется в виде $P = P_1 \| P_2 \| ... \| P_q$, $P_i \in V_s$, $i = 1, 2, ..., q-1$, $P_q \in V_r$, $r \le s$. Блоки шифр-текста вычисляются по следующему правилу:

$$
\begin{aligned}
&R_1 = IV,\\
&\quad Y_i = e_K(MSB_n(R_i)),\ i = 1, 2, ..., q-1,\\
&\quad C_i = P_i \oplus T_s(Y_i),\ i = 1, 2, ..., q-1,\\
&\quad R_{i+1} = LSB_{m-n}(R_i) \| Y_i,\ i = 1, 2, ..., q-1,\\
&Y_q = e_K(MSB_n(R_q)),\\
&C_q = P_q \oplus T_r(Y_q).
\end{aligned}
$$

Результирующий шифр-текст имеет вид $C = C_1 \| C_2 \| ... \| C_q$. Зашифрование в режиме гаммирования с обратной связью по выходу проиллюстрировано на рис. 78.

Расшифрование. Шифр-текст представляется в виде $C = C_1 \| C_2 \| ... \| C_q$, $C_i \in V_s$, $i = 1, 2, ..., q-1$, $C_q \in V_r$, $r \le s$. Блоки открытого текста вычисляются по следующему правилу:

$$
\begin{aligned}
&R_1 = IV,\\
&\quad Y_i = e_K(MSB_n(R_i)),\ i = 1, 2, ..., q-1,\\
&\quad P_i = C_i \oplus T_s(Y_i),\ i = 1, 2, ..., q-1,\\
&\quad R_{i+1} = LSB_{m-n}(R_i) \| Y_i,\ i = 1, 2, ..., q-1,\\
&Y_q = e_K(MSB_n(R_q)),\\
&P_q = C_q \oplus T_r(Y_q).
\end{aligned}
$$

Исходный открытый текст имеет вид $P = P_1||P_2||...||P_q$. Расшифрование в режиме гаммирования с обратной связью по выходу проиллюстрировано на рис. 79.

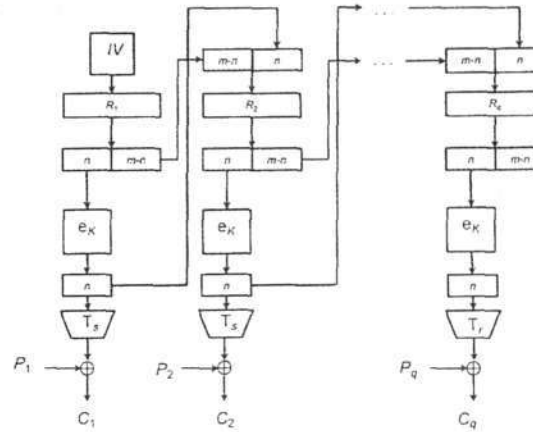


Рис. 78. Зашифрование в режиме гаммирования с обратной связью по выходу

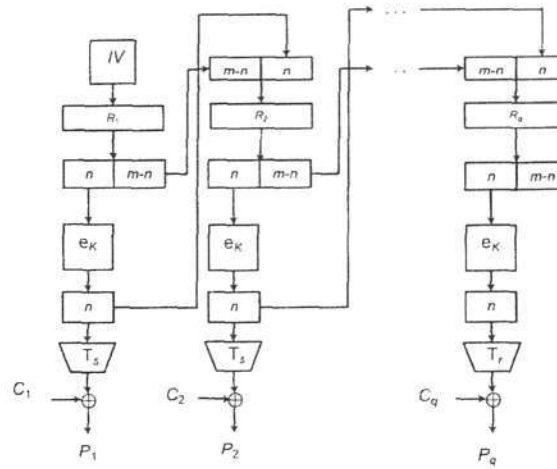


Рис. 79. Расшифрование в режиме гаммирования с обратной связью по выходу

5.3.5. Режим простой замены с зацеплением (СВС-ГОСТ)

Параметром режима простой замены с зацеплением является целочисленная величина m, $m = n \cdot z$, $z \geq 1$ – целое число. Длина сообщений, зашифровываемых в режиме простой замены с зацеплением, должна быть кратна длине блока базового алгоритма блочного шифрования n, поэтому, при необходимости, к исходному сообщению должна быть предварительно применена процедура дополнения. При шифровании на одном ключе для каждого отдельного открытого текста используется значение непредсказуемой (случайной или псевдослучайной) синхропосылки $IV \in V_m$. При шифровании в режиме простой замены с зацеплением используется двоичный регистр сдвига R длиной m.

Начальным заполнением регистра является значение синхропосылки IV. В режиме простой замены с зацеплением очередной блок шифр-текста получается путем зашифрования результата покомпонентного сложения значения очередного блока открытого текста со значением n разрядов регистра сдвига с большими номерами. Затем регистр сдвигается на один блок в сторону разрядов с большими номерами. В разряды с меньшими номерами записывается значение блока шифр-текста.

Зашифрование. Открытый и, при необходимости, дополненный текст $P \in V^*$, $|P| = n \cdot q$, представляется в виде $P = P_1 \| P_2 \| \ldots \| P_q$, $P_i \in V_n$, $i = 1, 2, \ldots, q$. Блоки шифр-текста вычисляются по следующему правилу:

$$R_1 = IV,$$
$$C_i = e_K(P_i \oplus MSB_n(R_i)),\ i = 1, 2, \ldots, q-1,$$
$$R_{i+1} = LSB_{m-n}(R_i) \| C_i,\ i = 1, 2, \ldots, q-1,$$
$$C_q = e_K(P_q \oplus MSB_n(R_q)).$$

Результирующий шифр-текст имеет вид $C = C_1 \| C_2 \| \ldots \| C_q$. Зашифрование в режиме простой замены с зацеплением проиллюстрировано на рис. 80.

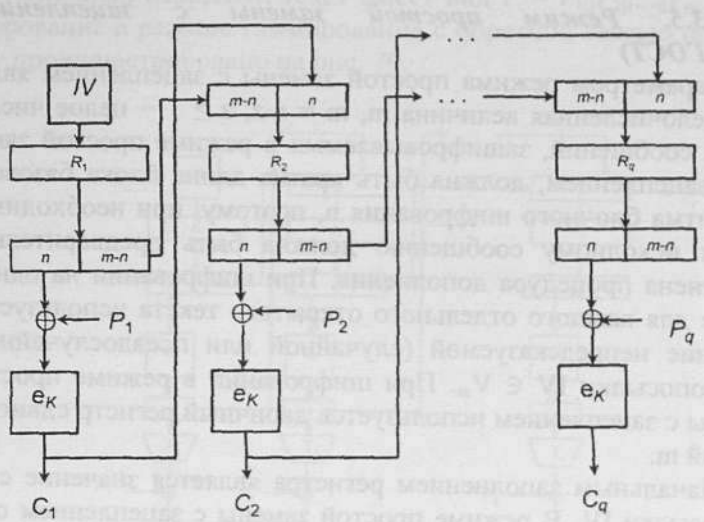


Рис. 80. Зашифрование в режиме простой замены с зацеплением

Расшифрование. Шифр-текст представляется в виде $C = C_1||C_2||...||C_q$, $C_i \in V_n$, $i = 1, 2, ..., q$. Блоки открытого текста вычисляются по следующему правилу:

$$R_1 = IV,$$
$$P_i = d_K(C_i) \oplus MSB_n(R_i),\ i = 1, 2, ..., q-1,$$
$$R_{i+1} = LSB_{m-n}(R_i)||C_i,\ i = 1, 2, ..., q-1,$$
$$P_q = d_K(C_q) \oplus MSB_n(R_q).$$

Исходный (дополненный) открытый текст имеет вид $P = P_1||P_2||...||P_q$. Если к исходному открытому тексту была применена процедура дополнения, то после расшифрования следует произвести обратную процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения. Расшифрование в режиме простой замены с зацеплением проиллюстрировано на рис. 81.

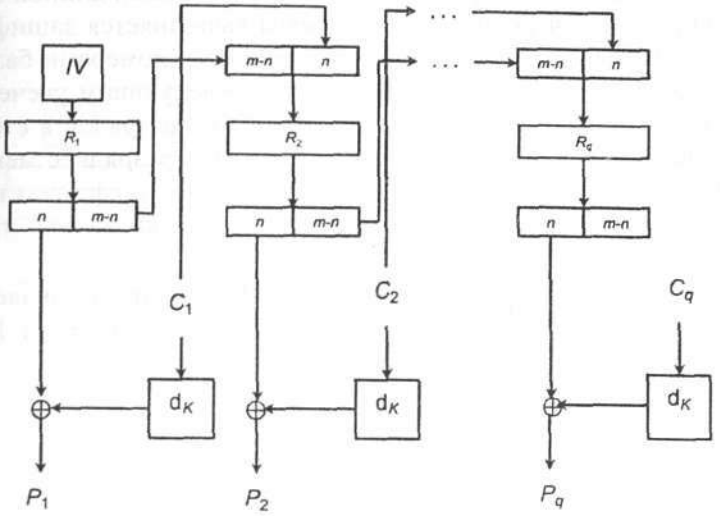


Рис. 81. Расшифрование в режиме простой замены с зацеплением

5.3.6. Режим гаммирования с обратной связью по шифр-тексту (CFB-ГОСТ)

Параметрами режима гаммирования с обратной связью по шифр-тексту являются целочисленные величины s и m, такие, что $0 < s \le n$, $n \le m$. В конкретной системе обработки информации на длину сообщения P может как накладываться ограничение $|P| = s \cdot q$, так и не накладываться никаких ограничений. В случае, если такое ограничение накладывается, к исходному сообщению, при необходимости, должна быть предварительно применена процедура дополнения. При шифровании на одном ключе для каждого отдельного открытого текста используется значение непредсказуемой (случайной или псевдослучайной) синхропосылки $IV \in V_m$. При шифровании в режиме гаммирования с обратной связью по шифр-тексту используется двоичный регистр сдвига R, длиной m. Начальным заполнением регистра является значение синхропосылки IV. Зашифрование в режиме гаммирования с обратной связью по шифр-тексту заключается в покомпонентном сложении открытого текста с

гаммой шифра, которая вырабатывается блоками длиной s. При вычислении очередного блока гаммы выполняется зашифрование n разрядов регистра сдвига с большими номерами базовым алгоритмом блочного шифрования с последующим усечением. Затем заполнение регистра сдвигается на s разрядов в сторону разрядов с большими номерами, при этом в разряды с меньшими номерами записывается полученный блок шифр-текста, являющийся результатом покомпонентного сложения гаммы шифра и блока открытого текста.

Зашифрование. Открытый текст $P \in V^*$ представляется в виде $P = P_1||P_2||...||P_q$, $P_i \in V_s$, $i = 1, 2, \ldots, q-1$, $P_q \in V_r$, $r \le s$. Блоки шифр-текста вычисляются по следующему правилу:

$$R_1 = IV,$$
$$C_i = P_i \oplus T_s(e_K(MSB_n(R_i))),$$
$$R_{i+1} = LSB_{m-s}(R_i)||C_i, \; i = 1, 2, \ldots, q-1,$$
$$C_q = P_q \oplus T_r(e_K(MSB_n(R_q))).$$

Результирующий шифртекст имеет вид $C = C_1||C_2||...||C_q$. Зашифрование в режиме гаммирования с обратной связью по шифртексту проиллюстрировано на рис. 82.

Расшифрование. Шифр-текст представляется в виде $C = C_1||C_2||...||C_q$, $C_i \in V_s$, $i = 1, 2, \ldots, q-1$, $C_q \in V_r$, $r \le s$. Блоки открытого текста вычисляются по следующему правилу:

$$R_1 = IV,$$
$$P_i = C_i \oplus T_s(e_K(MSB_n(R_i))), \; i = 1, 2, \ldots, q-1,$$
$$R_{i+1} = LSB_{m-s}(R_i)||C_i, \; i = 1, 2, \ldots, q-1,$$
$$P_q = C_q \oplus T_r(e_K(MSB_n(R_q))).$$

Исходный открытый текст имеет вид $P = P_1||P_2||...||P_q$. Если к исходному открытому тексту была применена процедура дополнения, то после расшифрования следует произвести обратную процедуру. Для однозначного восстановления сообщения может потребоваться знание длины исходного сообщения. Расшифрование в режиме гаммирования с обратной связью по шифр-тексту проиллюстрировано на рис. 83.

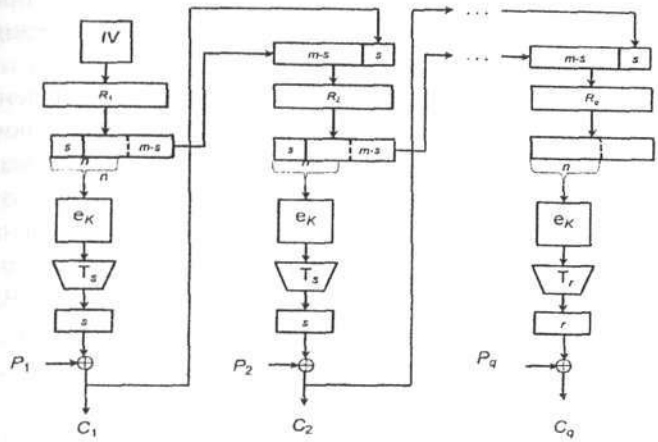


Рис. 82. Зашифрование в режиме гаммирования с обратной связью по шифр-тексту

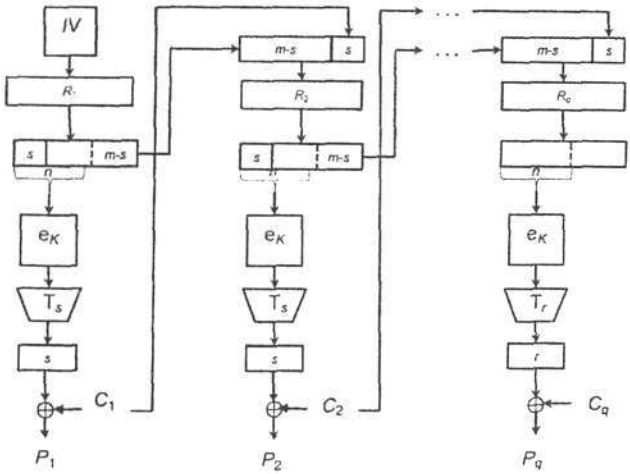


Рис. 83. Расшифрование в режиме гаммирования с обратной связью по шифр-тексту

5.3.7. Режим выработки имитовставки (MAC-ГОСТ)

Режим выработки имитовставки, описание которого представлено ниже, реализует конструкцию. Параметром режима является длина имитовставки (в битах) $0 < s \leq n$.

Выработка вспомогательных ключей. При вычислении значения имитовставки используются вспомогательные ключи, которые вычисляются с использованием ключа K. Длины вспомогательных ключей равны длине блока n базового алгоритма блочного шифрования. Процедура выработки вспомогательных ключей может быть представлена в следующей форме $R = e_K(0^n)$; $K_1 = R \ll 1$, если $MSB_1(R) = 0$ и $K_1 = (R \ll 1) \oplus B_n$ в противном случае; $K_2 = K_1 \ll 1$, если $MSB_1(K_1) = 0$ и $K_2 = (K_1 \ll 1) \oplus B_n$ в противном случае, где $B_{64} = 0^{59} \| 11011$, $B_{128} = 0^{120} \| 10000111$. Если значение n отлично от 64 и 128, следует использовать следующую процедуру определения значения константы B_n. Рассмотрим множество примитивных многочленов степени n над полем GF(2) с наименьшим количеством ненулевых коэффициентов. Упорядочим это множество лексикографически по возрастанию векторов коэффициентов и обозначим через $f_n(x)$ первый многочлен в этом упорядоченном множестве. Рассмотрим поле $GF(2^n)[x]/(f_n(x))$, зафиксируем в нем степенной базис и будем обозначать операцию умножения в этом поле символом $\otimes$. Вспомогательные ключи K_1 и K_2 вычисляются следующим образом:

$$R = e_K(0^n),$$
$$K_1 = Poly_n^{-1}(Poly_n(R) \otimes x),$$
$$K_2 = Poly_n^{-1}(Poly_n(R) \otimes x^2).$$

Вспомогательные ключи K_1, K_2 и промежуточное значение R наряду с ключом K являются секретными параметрами. Компрометация какого-либо из этих значений приводит к возможности построения эффективных методов анализа всего алгоритма.

Вычисление значения имитовставки. Процедура вычисления значения имитовставки похожа на процедуру зашифрования в режиме простой замены с зацеплением при m=n и ини-

циализации начального заполнения регистра сдвига значением 0^n: на вход алгоритму шифрования подается результат покомпонентного сложения очередного блока текста и результата зашифрования на предыдущем шаге. Основное отличие заключается в процедуре обработки последнего блока: на вход базовому алгоритму блочного шифрования подается результат покомпонентного сложения последнего блока, результата зашифрования на предыдущем шаге и одного из вспомогательных ключей. Конкретный вспомогательный ключ выбирается в зависимости от того, является ли последний блок исходного сообщения полным или нет. Значением имитовставки MAC является результат применения процедуры усечения к выходу алгоритма шифрования при обработке последнего блока. Исходное сообщение $P \in V^*$, для которого требуется вычислить имитовставку, представляется в виде $P = P_1||P_2||\ldots||P_q$, где $P_i \in V_n$, $i = 1, 2, \ldots, q-1$, $P_q \in V_r$, $r \leq n$. Процедура вычисления имитовставки описывается следующим образом:

$$C_0 = 0^n,$$
$$C_i = e_K(P_i \oplus C_{i-1}),\ i = 1, 2, \ldots, q-1,$$
$$MAC = T_s(e_K(P_q^* \oplus C_{q-1} \oplus K^*)),$$

где $K^* = K_1$, если $|P_q| = n$ и $K^* = K_2$ в противном случае, при этом P_q^* – последний блок сообщения, полученного в результате дополнения исходного сообщения с помощью процедуры 3. Процедура вычисления имитовставки проиллюстрирована на рис. 84 – 86.

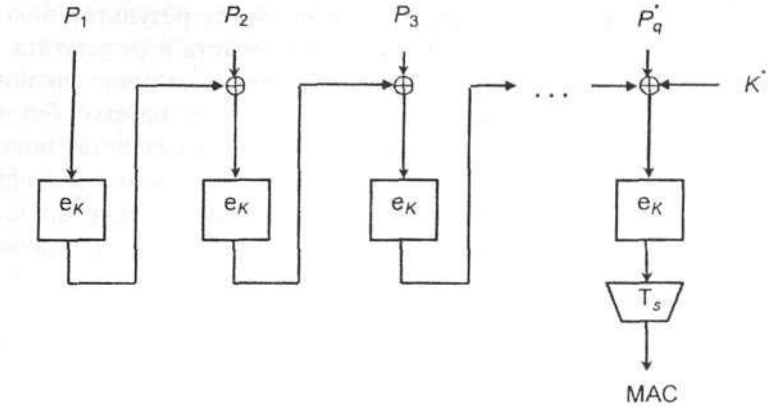


Рис. 84. Вычисление значения имитовставки (общий вид)

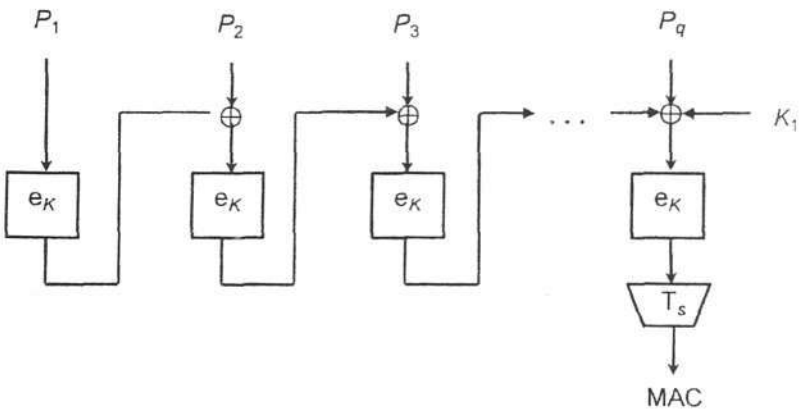


Рис. 85. Вычисление значения имитовставки (случай полного последнего блока)

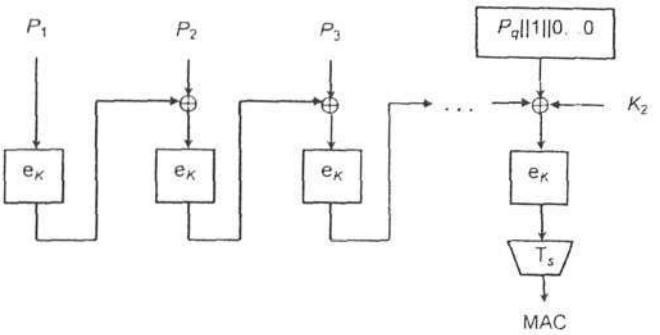


Рис. 86. Вычисление значения имитовставки
(случай с дополнением последнего блока)

6. ПОТОЧНЫЕ ШИФРЫ

По принципу преобразования информации современные симметричные криптосистемы можно разделить на два больших класса: блочные и поточные. Различные блочные алгоритмы шифрования, а также режимы их работы, мы рассмотрели в предыдущих разделах. В настоящем разделе мы ознакомимся с основной идеей поточных шифров. Поточные шифры оперируют с битами (реже с байтами), стараясь обеспечить шифрование в режиме реального времени или близком к нему. Высокая скорость работы поточных шифров определяет область их использования – закрытие данных, требующих оперативной доставки потребителю. Например, когда вы разговариваете по закрытому каналу связи, то весь потом аудиоинформации переводится в цифровой вид, зашифровывается и передается по каналу связи в зашифрованном виде. На приемной стороне поток расшифровывается и восстанавливается.

По принципу работы поточные шифры делят на комбинированные, синхронные и самосинхронизирующиеся. Комбинированные методы шифрования (а точнее, поточные режимы использования блочных шифров) используют принцип формирования потока ключей (гаммы шифра) с помощью генераторов псевдослучайных последовательностей, в качестве функции обратной связи или функции выхода которых используется функция зашифрования блочного шифра. Как правило, для комбинированных методов шифрования используются блочные шифры, являющиеся государственными стандартами шифрования данных: ГОСТ 28147-89 (российский стандарт) и AES (стандарт США).

В синхронных поточных шифрах гамма формируется независимо от входной последовательности, каждый элемент (бит, символ, байт) которой таким образом шифруется независимо от других элементов. В синхронных поточных шифрах отсутствует эффект размножения ошибок, т. е. число искаженных элементов в расшифрованной последовательности равно числу искаженных элементов зашифрованной последовательности,

пришедшей из канала связи. Вставка или выпадение элемента зашифрованной последовательности недопустимы, так как из-за нарушения синхронизации это приведет к неправильному расшифрованию всех последующих элементов. К синхронным поточным шифрам в первую очередь относится алгоритм A5, используемый в системах GSM (Group Special Mobile) для закрытия связи между абонентом и базовой станцией. Кроме того, к синхронным шифрам относится достаточно большое число алгоритмов, среди них: ORYX, PIKE, RC4, SEAL, CHAMELEON, PANAMA, TWOPRIME, SOBER, LILI-128, SNOW, LEVITHAN, SOLITAIRE, MIRDEK, SQ 1-R, COS.

В самосинхронизирующихся поточных шифрах элементы входной последовательности зашифровываются с учетом N предшествующих элементов, которые принимают участие в формировании ключевой последовательности. В самосинхронизирующихся шифрах имеет место эффект размножения ошибок, в то же время в отличие от синхронных восстановление синхронизации происходит автоматически через N элементов зашифрованной последовательности. К самосинхронизирующимся шифрам относятся такие шифры, как WAKE, схема Маурера, SAPPHIRE II.

Как правило, анализ поточных шифров сводится к анализу математических и статистических свойств генератора гаммы. Порождаемая, известная криптоаналитику гамма должна обладать следующими основными свойствами: исключать восстановление ключа шифра, не допускать предсказание гаммы по известному ее отрезку (как вперед, так и назад), ничем не отличаться от случайной последовательности, иметь максимально возможный период.

Для поточных шифров большое внимание уделяется исследованию свойств вырабатываемой гаммы от истинно случайной последовательности, имеющей распределение Бернулли. Для нахождения отличий можно использовать самые разнообразные статистические тесты.

6.1. Регистры сдвига с обратной линейной связью

Для построения поточных шифров очень часто используют последовательности на регистрах сдвига. Регистр сдвига с обратной связью состоит из двух частей: регистра сдвига и функции обратной связи, как показано на рис. 87. Сам регистр сдвига представляет собой последовательность битов, число которых определяет длину регистра. Так, если в регистре задействовано n бит, то говорят, что n-битовый регистр сдвига. При каждом извлечении бита все биты регистра сдвига сдвигаются вправо на одну позицию, обычно в сторону младших разрядов. Периодом регистра сдвига называют длину получаемой последовательности до начала ее повторения [2].

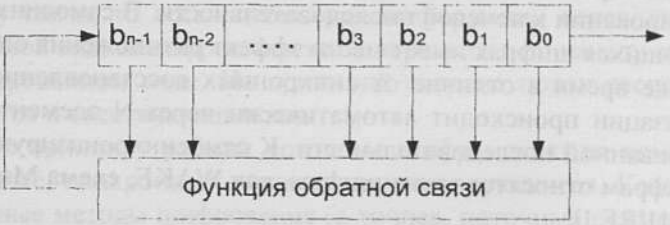


Рис. 87. Регистр сдвига с обратной линейной связью

В качестве обратной связи может выступать любая математическая функция, производящая действие над битами. К простейшему типу регистра сдвига с обратной связью относится регистр сдвига с линейной обратной связью (РСЛОС) [2]. В РСЛОС обратная связь представляет собой просто операцию сложения по модулю 2 (XOR) над некоторыми битами регистра; перечень этих битов называется последовательностью отводов, или точек съема, как показано на рис. 88. Иногда такую схему называют конфигурацией Фибоначчи [2]. Благодаря простоте последовательности обратной связи, для анализа РСЛОС можно использовать довольно развитую математическую теорию. В любом случае качество вырабатываемой ПСП оценивают специальным набором тестов.

Рис. 88. Регистр сдвига с линейной обратной связью (РСЛОС)

Записываются РСЛОС в виде полиномов. При этом степень первого элемента полинома указывает на количество битов в регистре сдвила, а степенные показатели остальных членов полинома показывают, какие будут использованы точки съема. Так, например, запись $x^4 + x + 1$ обозначает, что будет использован регистр из четырех элементов, для которого в образовании обратной связи будут участвовать биты b_1 и b_0 (рис. 89).

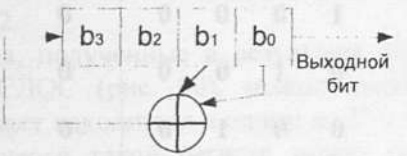


Рис. 89. 4-битовый РСЛОС

Рассмотрим работу регистра, представленного на рис. 89. Инициализируем его, например, значением 0101 (начальная инициализация может выполняться любой последовательностью битов, кроме последовательности из одних нулей, так как в таком случае обратная связь всегда будет образовывать значение ноль и регистр не будет вырабатывать ожидаемую ПСП). Итак, в регистре происходит сдвиг вправо на одну позицию. Самый младший бит, равный 1, вытесняется из регистра и образует первый бит ПСП. Те биты, которые были в позициях b_1 и b_0, до сдвига складываются по модулю 2 и образуют новый

старший бит регистра. Наглядный пример работы рассматриваемого РСЛОС приведен на рис. 90.

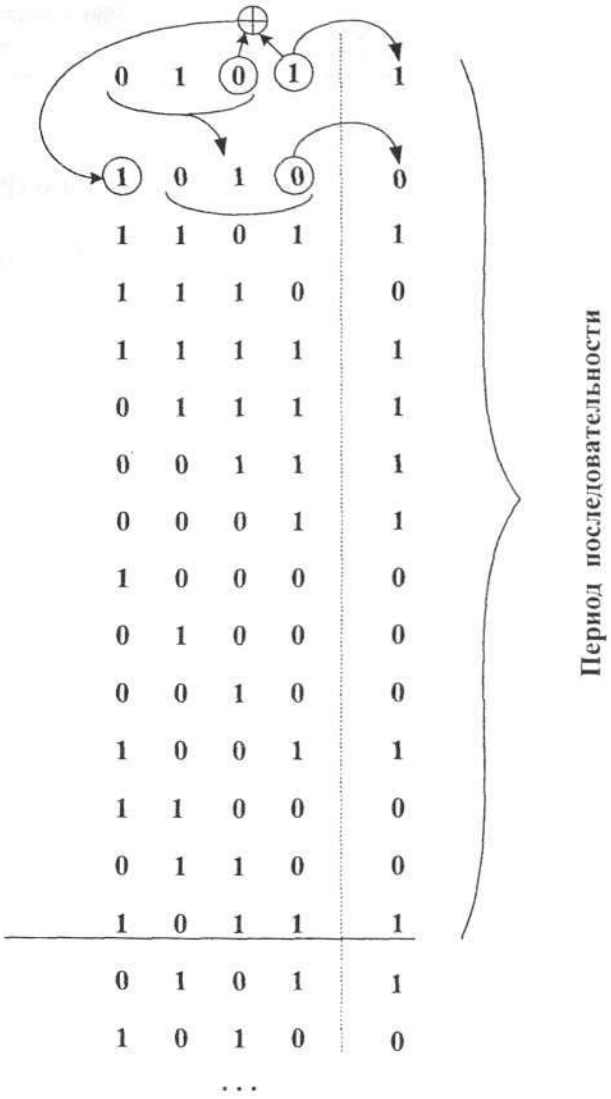


Рис. 90. Пример работы 4-битового РСЛОС

Как видно из рис. 90, заполнение регистра пройдет через все 15 из 16 возможных состояний (ранее мы определили, что шестнадцатое состояние, когда РСЛОС равен 0000, рассматривать нельзя). После этого заполнение регистра станет опять равно начальному значению 0101, и выработка битовой последовательности начнет повторяться. Выходной последовательностью регистра будет строка младших значащих битов (до горизонтальной линии на рис. 90): 101011110001001. Размер битовой последовательности до ее повторения называется периодом. Для обеспечения максимального периода конкретного РСЛОС (т. е. для того, чтобы регистр прошел через все возможные внутренние состояния) многочлен, который определяет работу регистра, должен быть примитивным по модулю 2. Как и в случае с большими простыми числами, не существует способа генерации таких многочленов. Можно лишь взять многочлен и проверить, является он неприводимым по модулю 2 или нет. В общем случае примитивный многочлен степени n – это такой неприводимый многочлен, который является делителем $x^{2^n-1}+1$, но не является делителем x^d+1 для всех d, являющихся делителями 2^n-1 [2]. В работе Б. Шнайера [2] приведена таблица некоторых многочленов, которые являются неприводимыми по модулю 2.

Обобщая знания, полученные в результате рассмотрения примера работы РСЛОС (рис. 90), можно сказать, что n-битовый РСЛОС может находиться в одном из 2^n-1 внутренних состояний. Теоретически такой регистр может генерировать псевдослучайную последовательность с периодом 2^n-1 битов. Такие регистры называются регистрами РСЛОС с максимальным периодом. Получившийся выход называют m-последовательностью [2].

6.2. Алгоритм A5/1

Шифр A5/1 – это поточный шифр, используемый для шифрования связи GSM (Group Special Mobile – мобильная групповая специальная связь). Это европейский стандарт для мобильных цифровых сотовых телефонов. Он используется для шифрования канала «телефон/базовая станция».

С шифром A5/1 не все так просто, как с описанием остальных стандартов. Долгое время этот шифр был засекречен и общедоступным стал после утечки информации. Однако до сих пор отсутствует официальное описание схемы работы шифра A5/1. Изучая различные источники, можно запутаться. Потому как разные ученые по-разному изображают и сам шифр, и полиномы, лежащие в его основе, так же по-разному отображают используемые точки синхронизации. Проведя широкий анализ различных публикаций, мы убедились в том, что в основе всех описаний лежит одна и та же схема работы, однако описывают ее по-разному. Попробуем прояснить картину и разберем, как же работает алгоритм шифрования A5/1.

Генератор A5/1 состоит из трех РСЛОС длиной 19, 22 и 23, все многочлены обратной связи у него являются неприводимыми по модулю 2 (см. подразд. 6.1). Выходом является результат операций XOR над тремя РСЛОС. В A5/1 используется изменяемое управление тактированием. Каждый регистр тактируется в зависимости от своего среднего бита, затем над регистром выполняется операция XOR с обратной пороговой функцией средних битов всех трех регистров.

Каждый кадр шифруется с помощью секретного ключа шифрования *Kc* и сквозного порядкового номера очередного кадра. Генератор ПСП A5/1 состоит из трех коротких РСОЛС, обозначаемых как РСОЛС 1, РСОЛС 2 и РСОЛС 3. Выходные биты снимаются с самых старших разрядов регистров, после чего с помощью операции XOR над битами с выходов всех трех регистров формируется выходной бит γ шифра. Общий вид работы алгоритма шифрования A5/1 представлен на рис. 91.

Итак, согласно Б. Шнайеру [2], РСЛОС обычно производят сдвиг в сторону младших разрядов. Однако в случае с шифром A5/1 дело обстоит иначе, сдвиг происходит в сторону старших разрядов. На рис. 91 изображены три РСЛОС. Слева в них находятся старшие значащие биты, справа – младшие. Для того чтобы понять соответствие различных описаний шифра A5/1, мы представили на рис. 91 нумерацию битов как слева направо, начиная от нуля, так и справа налево, также от нуля.

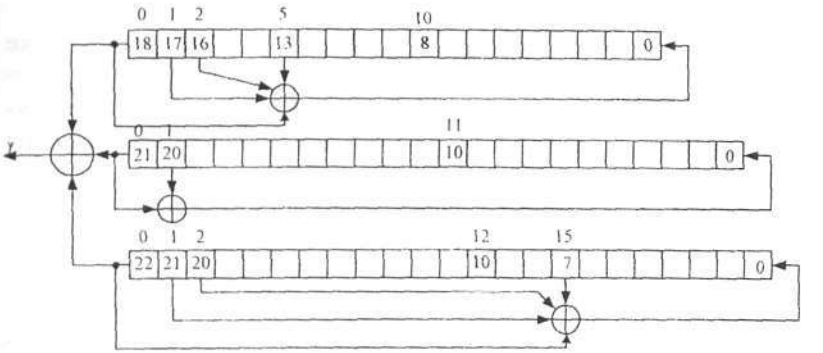


Рис. 91. Алгоритм шифрования A5/1

Из рис. 91 видно, что многочлены, лежащие в основе шифра A5/1, можно описать двумя разными способами. Если следовать логике, предложенной Б. Шнайером, то, используя нумерацию слева направо, начиная от нуля, получится следующая картина:

РСОЛС 1: $x^{19} + x^5 + x^2 + x + 1$;
РСОЛС 2: $x^{22} + x + 1$;
РСОЛС 3: $x^{23} + x^{15} + x^2 + x + 1$.

При этом первые два многочлена можно найти в таблице неприводимых полиномов в книге Б. Шнайера [2]. Третий регистр также является неприводимым по модулю 2. Согласно [2], полином, который в данном поле является образующим, называется примитивным, или базовым, если все его коэффициенты являются взаимно простыми. Для многочлена, описывающего РСЛОС3, это условие выполняется.

Регистры шифра A5/1 работают по принципу stop-and-go, что обеспечивается с помощью применения специальной функции majority, на вход которой подаются значения битов регистров: бит C1 для РСОЛС 1, бит C2 для РСОЛС 2 и C3 для РСОЛС 3. В некоторых источниках указывается следующее соответствие битов: C1 = 8, C2 = 10, C3 = 10. Это соответствует нумерации справа налево, начиная от нуля, при этом порядко-

вые номера битов будут соответственно 9, 11 и 11. В других же источниках можно встретить указание на C1 = 11, C2 = 12, C3 = 13, что соответствует тем же самым битам, что и в первом случае, но при нумерации слева направо и начиная не от нуля, а от единицы.

Функция majority имеет следующий вид:

$$\text{Majority}(x1,x2,x3) = x1x2+x1x3+x2x3.$$

Результатом работы этой функции является один бит, который определяет, какие регистры будут тактироваться (сдвигаться), а какие нет. То есть, если бит с выхода функции совпадает со значением бита регистра на определенной позиции, то регистр сдвигается, иначе нет. Фактически эта функция является функцией большинства, она принимает то значение, которое преобладает на ее входах. В любом случае в режиме работы stop-and-go минимум два регистра будут работать на сдвиг (те, для которых определяющий бит оказался в большинстве). А в случае, когда определяющий бит совпадает у всех регистров, тактироваться будут все три регистра. Работа регистров по правилу stop-and-go обеспечивает нелинейность работы алгоритма в целом, схема работы этого режима схематично показана на рис. 92. Функцию определения, какой из регистров необходимо тактировать, а какой нет, выполняет блок управления синхронизацией регистров (Clock Control).

На каждом шаге работы шифра два или три регистра сдвигаются. Таким образом, каждый регистр сдвигается в одном такте работы алгоритма с вероятностью ¾ и не сдвигается с вероятностью ¼.

Процесс формирования гаммы, необходимой для шифрования одного кадра, состоит из следующих шагов.

1. Все три регистра сбрасываются в ноль, а затем тактируются 64 раза без учета режима stop-and-go (все три регистра тактируются). Во время этого этапа каждый бит ключа *Kc* последовательно записывается в самый младший бит каждого регистра после операции XOR с сигналом обратной связи, как показано на рис. 93.

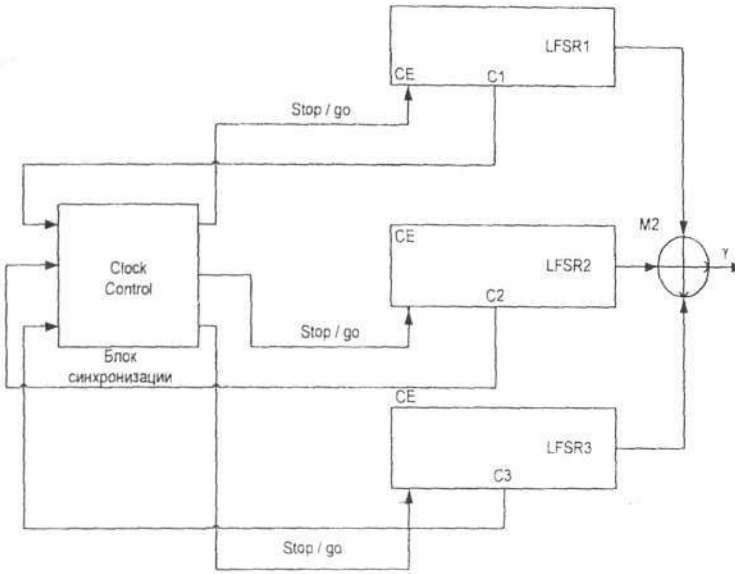


Рис. 92. Схема работы регистров по правилу stop-and-go в алгоритме A5/1

2. Все три регистра тактируются 22 раза без учета режима stop-and-go. Во время этого этапа биты номера кадра (так же как и биты ключа шифрования Кс в предыдущем пункте) с помощью операции XOR с сигналом обратной связи последовательно записываются в самый младший разряд каждого регистра.

3. Осуществляется 100 тактов работы алгоритма с использованием режима stop-and-go, что обеспечивает перемешивание ключевой информации.

4. Осуществляется 228 тактов работы алгоритма с использованием режима stop-and-go для формирования гаммы. Затем осуществляется шифрование, когда с помощью операции XOR сформированная последовательность накладывается на информацию, содержащуюся в кадре [1].

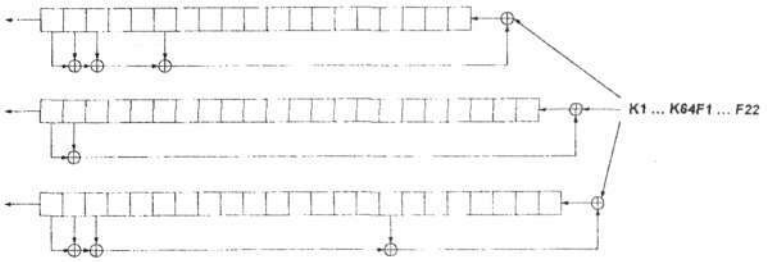


Рис. 93. Инициализация ключа К и номера кажра F

6.3. Усеченная модель алгоритма А5/У

В настоящем подразделе мы рассмотрим усеченную модель алгоритма А5/У, в которой соблюдены все пропорции исходного шифра. Эта модель является более простой и удобной для практического изучения основных принципов построения поточных шифров. Благодаря малой длине регистров и ключа она не требует столько времени на обработку параметров, как сам алгоритм А5/1.

Каждый кадр шифруется с помощью секретного ключа шифрования *Кс* и сквозного порядкового номера очередного кадра. Генератор ПСП А5/1 состоит из трех коротких РСОЛС, как показано на рис. 94, обозначаемых как РСОЛС 1, РСОЛС 2 и РСОЛС 3. Образующие многочлены этих регистров имеют вид

$$\text{РСОЛС 1: } x^6 + x + 1;$$
$$\text{РСОЛС 2: } x^7 + x3 + 1;$$
$$\text{РСОЛС 3: } x^{11} + x2 + 1.$$

Выходные биты снимаются с самых старших разрядов регистров, после чего с помощью операции XOR над битами с выходов всех трех регистров формируется выходной бит γ шифра. Регистры работают по принципу stop-and-go, что обеспечивается с помощью применения специальной функции majority, на вход которой подаются значения битов регистров: бит С1 (третий разряд) для РСОЛС 1, бит С2 (пятый разряд) для РСОЛС 2 и С3 (пятый разряд) для РСОЛС 3.

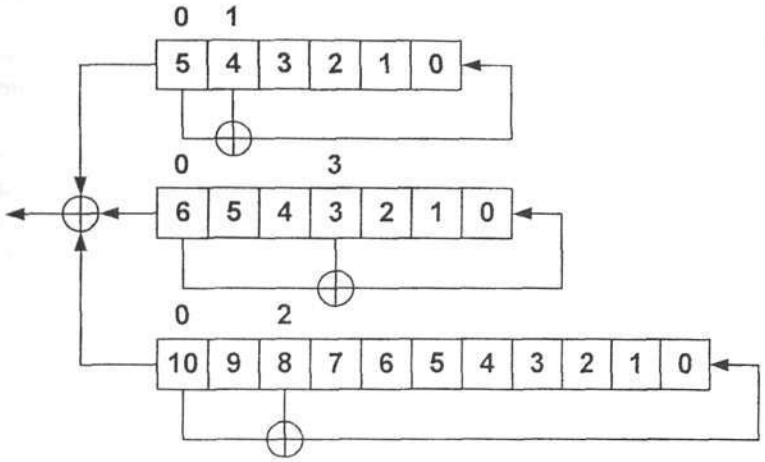


Рис. 94. Схема работы регистров усеченного алгоритма A5

Функция majority имеет следующий вид:

$$\text{Majority}\,(x1,x2,x3) = x1x2+x1x3+x2x3.$$

Результатом работы этой функции является один бит, который определяет, какие регистры будут тактироваться (сдвигаться), а какие нет, аналогично тому, как это делается для оригинального алгоритма A5/1 (см. подраздел 6.2.)

Процесс формирования гаммы, необходимой для шифрования одного кадра, состоит из следующих шагов.

1. Все три регистра сбрасываются в ноль, а затем тактируются 24 раза без учета режима stop-and-go (все три регистра тактируются). Во время этого этапа каждый бит ключа Kc последовательно записывается в самый младший бит каждого регистра после операции XOR с сигналом обратной связи, как показано на рис. 92.

2. Все три регистра тактируются 8 раз без учета режима stop-and-go. Во время этого этапа биты номера кадра, так же как и биты ключа шифрования Kc с помощью операции XOR с сиг-

налом обратной связи последовательно записываются в самый младший разряд каждого регистра.

3. Осуществляется 32 такта работы алгоритма с использованием режима stop-and-go, что обеспечивает перемешивание ключевой информации.

4. Осуществляется 32 такта работы алгоритма с использованием режима stop-and-go для формирования гаммы. Затем осуществляется шифрование, когда с помощью операции XOR сформированная последовательность накладывается на информацию, содержащуюся в кадре.

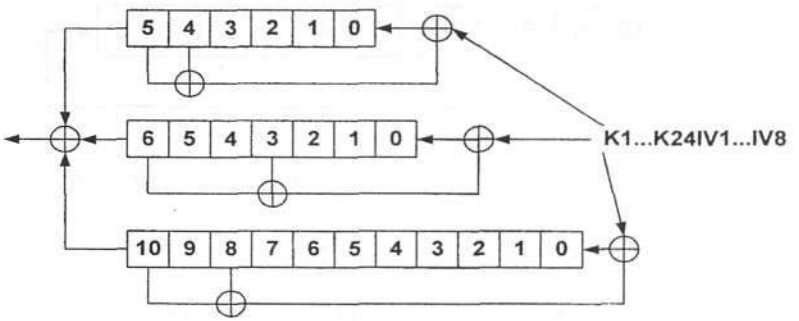


Рис. 95. Инициализация ключа и вектора IV
в усеченном алгоритме А5/У

7. МАЛОРЕСУРСНАЯ КРИПТОГРАФИЯ

В последнее время наблюдается тенденция массового перехода от интернета персональных компьютеров к интернету вещей. В нашу жизнь все прочнее и прочнее входит бесчисленное число всевозможных электронных приспособлений, направленных на улучшение жизни и так или иначе взаимодействующих с интернетом. Айфоны, айпады, телефоны, электронные ключи, бытовая техника – все это снабжено процессором и функционирует в составе большой сети Интернет. Даже холодильник, и тот, например, может иметь выход в интернет и посылать запрос в интернет-магазин о доставке необходимых продуктов. Все это многообразие порождает так называемый интернет вещей. Интернет вещей (Internet of Things, IoT) представляет собой беспроводную самоконфигурирующуюся сеть между объектами типа бытовых приборов, транспортных средств, датчиков, а также меток радиочастотной идентификации (Radio Frequency IDentification, RFID) [14].

В связи с тем, что многие приспособления мобильны и предназначены для того, чтобы человек всегда имел их при себе, зачастую они имеют маленькие размеры и как следствие ограничены в ресурсе по времени и памяти, имеют ограничения в вычислительной мощности, а время работы их зависит от надежности аккумулятора. По данным исследования, уже сейчас во всем мире 98,8 % всех изготовленных процессоров используется во встроенных приложениях различных приспособлений и лишь 1,2 % используется в составе персональных вычислительных машин.

Легковесная криптография – раздел криптографии, имеющий своей целью разработку алгоритмов для применения в устройствах, которые не способны обеспечить большинство существующих шифров достаточными ресурсами (память, электропитание, размеры) для функционирования.

Из того арсенала, который есть в наличии легковесной современной криптографии, всего два шифра, которые в 2012 г. были включены вмеждународный стандарт облегченного шиф-

рования ISO/IEC 29192-2:2012. Это алгоритмы шифрования Present и Clefia. Шифр Present представляет собой легкий блочный шифр с размером блока данных 64 бита и размером ключа 80 бит для PRESENT-80 или 128 бит для PRESENT-128. Шифр Clefia представляет собой легкий блочный шифр с размером блока данных 128 бит и размером ключа 128, 192 или 256 бит.

Помимо этих двух шифров, также широко известен шифр Trivium, который является симметричным поточным шифром и ориентирован в первую очередь на аппаратную реализацию.

Именно эти три шифра будут рассмотрены в настоящем разделе.

7.1. Алгоритм шифрования CLEFIA

Алгоритм шифрования CLEFIA представляет собой 128-битовый блочный шифр с длиной ключа 128, 192, 256 битов. Для обработки входного сообщения (рис. 96) использует обобщенную сеть Фейстеля с четырьмя ветвями. При этом в одном раунде параллельно применяются две функции F: F0 и F1 за раунд. Число раундов r для шифра CLEFIA-128 равно 18 [15]. На вход функции шифрования поступают 128 битов открытого текста $P = P_0|P_1|P_2|P_3$, 32-битные ключи зашумления WK_i ($0 \le i < 4$) и 32-битные раундовые подключи RK_j ($0 \le j < 2r$). На выходе функции образуется 128-битный зашифрованный текст $C = C_0|C_1|C_2|C_3$, как показано на рис. 96.

В каждом раунде шифрования одновременно используются две разные функции F. Эти две F-функции F0 и F1 состоят из сложения с раундовым подключом, четырех нелинейных преобразований с использованием 8-битных S-блоков, а также матрицы распространения. Принцип работы функций F0 и F1 показан на рис. 97. При этом используются два разных S-блока. Логика работы S-блоков показана на рис. 98. Матрицы распространения также различны для функций F0 и F1. Их обозначают как M0 для функции F0 и M1 для функции F1. Данные матрицы имеют следующий вид:

$$M_0 = \begin{pmatrix} 01 & 02 & 04 & 06 \\ 02 & 01 & 06 & 04 \\ 04 & 06 & 01 & 02 \\ 06 & 04 & 02 & 01 \end{pmatrix}, \quad M_1 = \begin{pmatrix} 01 & 08 & 02 & 0A \\ 08 & 01 & 0A & 02 \\ 02 & 0A & 01 & 08 \\ 0A & 02 & 08 & 01 \end{pmatrix}.$$

Умножение между матрицами и векторами выполняется в поле Галуа GF (2^8) и определяется примитивным полиномом $z^8 + z^4 + z^3 + z^2 + 1$.

На вход функции выработки раундовых подключей алгоритма шифрования CLEFIA-128 поступает секретный ключ K, 32-битные ключи зашумления WK_i ($0 \le i < 4$) и 32-битные раундовые подключи RK_j ($0 \le j < 2r$). Функция выработки подключей подразделяется на два этапа: выработку 128-битного промежуточного ключа L (шаг 1) и выработку значений WK_i и RK_j из значений K и L (шаг 2).

На шаге 1 для выработки промежуточного ключа L используются 12 раундов шифрования, на вход которых в качестве открытого текста поступает знание ключа K, а также постоянные величины CON_i ($0 \le i < 24$) в качестве раундовых подключей. На шаге 2 промежуточный ключ L обрабатывается с использованием функции DoubleSwap. Раундовые подключи RK_j ($0 \le j < 36$) вырабатываются путем перемешивания значений K, L и констант CON_i ($24 \le i < 60$). Ключам забеливания WK_i соответствуют 32-битные фрагменты K_i из ключа $K = K_0|K_1|K_2|K_3$.

Простая линейная операция под названием DoubleSwap (Двойной обмен) используется в функции выработки подключей. На рис. 99 показан принцип работы функциия DoubleSwap. На вход операции поступает 128-битовое значение, которое разделяется на 4 части, после чего эти части меняются местами. Функция DoubleSwap используется для того, чтобы быстро вырабатывать раундовые подключи, поддерживая при этом эффективность внедрения программного и аппаратного обеспечения [16].

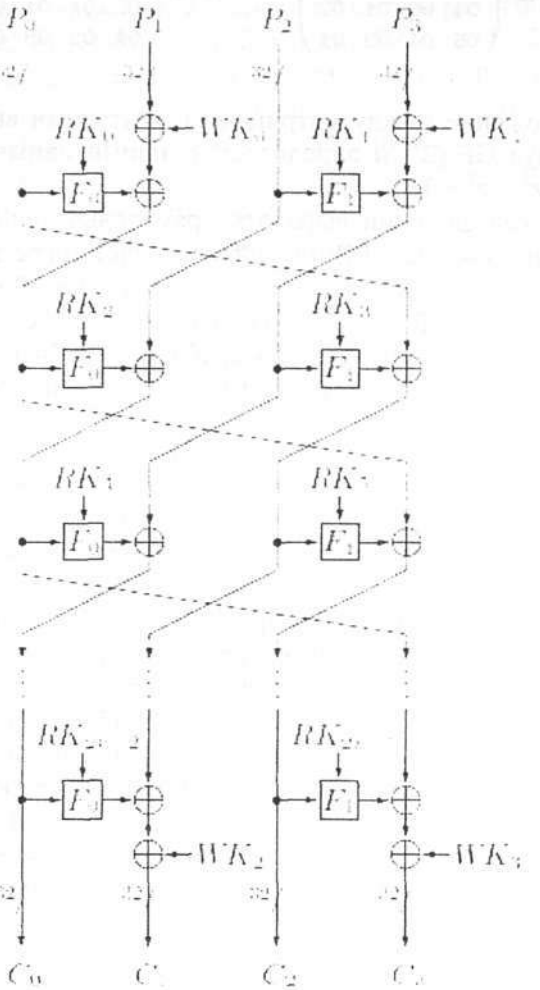


Рис. 96. Функция шифрования для алгоритма шифрования CLEFIA

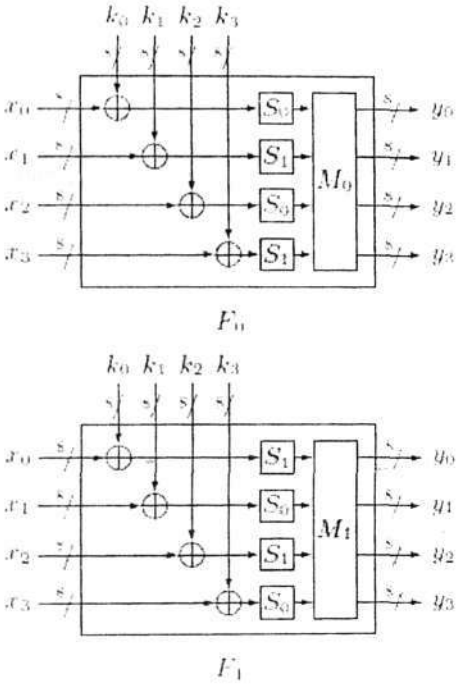


Рис. 97. Функции F_0 и F_1

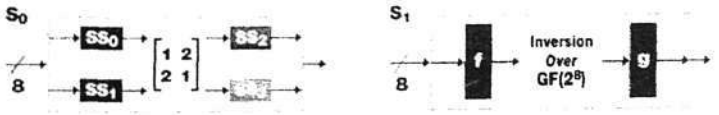


Рис. 98. Блоки замены S_0 и S_1

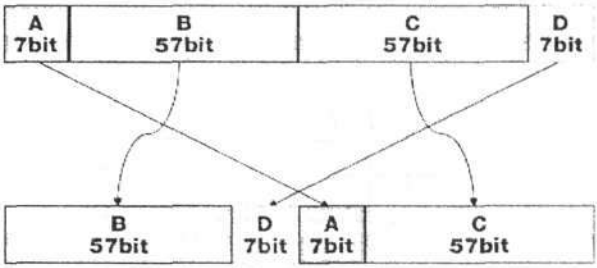


Рис. 99. Функция DoubleSwap

7.2. Алгоритм шифрования Present

Алгоритм шифрования Present представляет собой блочный шифр, обрабатывающий за один раз блок данных длиной 64 бита. Работа алгоритма выполняется в течение 31 раунда. Шифр имеет структуру SP-сети. При этом возможно использование двух разных ключей: длиной в 80 или 128 бит.

Каждый раунд состоит из выполнения трех основных преобразований: сложения данных с ключом по модулю 2, замены значений с использованием S-блока и операции перестановки.

Принцип работы S-блока замены отражен в табл. 37. Преобразование происходит так же, как это сделано для S-блоков замены в алгоритме шифрования ГОСТ 28147-89. Входное 4-битное значение на входе S-блока замены указывает на номер ячейки в таблице замены. Значение, стоящее в соответствующей ячейке таблицы, является выходом S-блока.

Таблица 37

Таблица S-box. Замена 4-битных блоков

Вход	0	1	2	3	4	5	6	7	8	A	B	C	D	E	F
Выход	C	5	6	B	9	0	A	D	3	E	F	8	4	7	2

Операция перестановки выполняется так же, как было рассмотрено для алгоритма шифрования DES, в соответствии с табл. 38. Первый бит останется на своем месте, пятый бит станет вторым, девятый – третьим и т.д. Если внимательно посмотреть на рис. 100. На котором изображено преобразование

двух раундов с помощью алгоритма Present, можно увидеть, что после перестановки первые 16 битов образуются первыми битами из выходов каждого S-блока, вторые 16 битов – вторыми битами выходов и т.д.

Таблица 38

Таблица перестановки для алгоритма шифрования Present

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

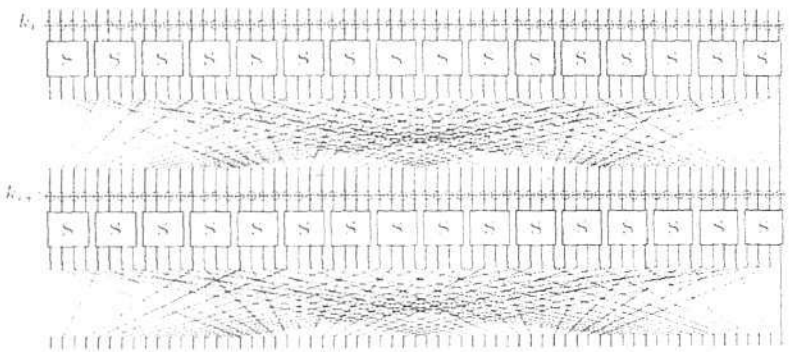


Рис. 100. Два раунда работы алгоритма шифрования Present

В алгоритме шифрования Present могут быть использованы два варианта ключа: длиной 80 и 128 бит. В каждом раунде происходит обновление ключа по следующим правилам.

Для 80-битных ключей:

Шаг 1. $[k_{79},k_{78}\ldots k_1,k_0] = [k_{18},k_{17}\ldots k_1,k_0,k_{79},k_{78}\ldots k_{20},k_{19}]$

Шаг 2. $[k_{79},k_{78},k_{77},k_{76}] = S[k_{79},k_{78},k_{77},k_{76}]$

Шаг 3. Последовательность $[k_{19},k_{18},k_{17},k_{16},k_{15}]$ складывается по модулю 2 с номером раунда

Для 128-битных ключей:

Шаг 1. $[k_{127},k_{126}\ldots k_1,k_0] = [k_{66},k_{65}\ldots k_{68},k_{67}]$

Шаг 2. $[k_{127},k_{126},k_{125},k_{124}] = S[k_{127},k_{126},k_{125},k_{124}]$

Шаг 3. $[k_{123},k_{122},k_{121},k_{120}] = S[k_{123},k_{122},k_{121},k_{120}]$

Шаг 4. Последовательность $[k_{66}, k_{65}, k_{64}, k_{63}, k_{62}] = [k_{66}, k_{65}, k_{64}, k_{63}, k_{62}]$ складывается по модулю 2 с номером раунда

Расшифрование данных происходит в обратном порядке, т. е. сначала входной блок складывается по модулю 2 с тридцать вторым раундовым подключом, далее проходит через инверсную таблицу перестановки и инверсный блок замены. Обработка выполняется в течение тридцати одного раунда, раундовые подключи используются в обратном порядке. То есть расшифрование завершается сложением данных по модулю 2 с первым раундовым подключом.

7.3. Алгоритм шифрования Trivium

Алгоритм шифрования Trivium представляет собой поточный шифр, т. е. при работе этого алгоритма генерируется псевдослучайная последовательность битов, которая, складываясь побитно по модулю 2 с соответствующими битами шифруемого текста, образует его шифр-текст.

Структура Trivium представляет собой три сдвиговых регистра общей длиной 288 бит с нелинейной комбинацией прямой и обратной связи, которые совместно формируют псевдослучайную последовательность (бегущий ключ) длиной вплоть до 2^{64} бит. При каждом такте выполнения алгоритма происходит сдвиг вправо на один бит в каждом из битовых регистров, а на выходе генерируется один бит бегущего ключа. Шифрование сообщения производится путем выполнения операции сложения по модулю 2 бит сообщения с соответствующими битами бегущего ключа. Расшифрование выполняется путем сложения по модулю 2 бит шифра сообщения с соответствующими битами бегущего ключа.

На рис. 101 представлен общий вид алгоритма Trivium. Числами представлена позиция бита в регистрах, операция $\otimes$ означает умножение, а операция $\oplus$ сложение по модулю 2.

Для начального заполнения регистров Trivium используются два источника: секретный ключ K и управляющий вектор IV. Размеры ключа и управляющего вектора составляют 80 бит.

Первый регистр алгоритма Trivium имеет длину 93 бита, второй – 84 бита, третий – 111 бит.

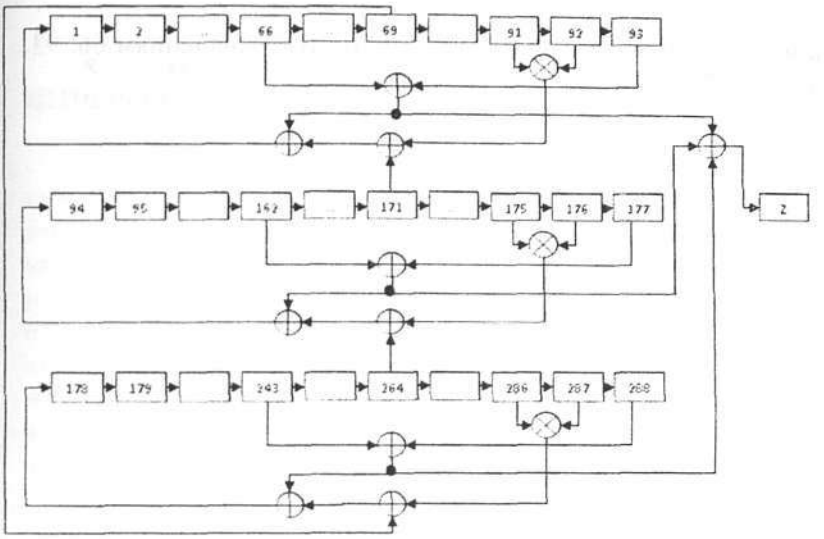


Рис. 101. Алгоритм шифрования Trivium

Перед генерацией псевдослучайной последовательности Z, необходимо произвести процедуру инициализации, заключающуюся в том, что вначале регистры Trivium заполняются битами из секретного ключа K и управляющего вектора IV. Поле этой процедуры происходит выполнение четырех циклов длиной в 288*4 тактов алгоритма Trivium, без генерации выходной последовательности Z. Последняя процедура необходима для того, чтобы каждый бит начального состояния регистров зависел от каждого бита вектора IV и ключа K. На самом деле это достигается при выполнении двух полных циклов (288*2 такта), остальные два цикла необходимы для усложнения взаимосвязей битов регистра.

В целом ход работы Trivium не слишком изменяется по отношению к процессу инициализации. Единственным отличием является то, что теперь на каждом такте выполнения алгоритма генерируется один бит последовательности Z, который представляет собой сумму по модулю 2 шести битов регистров (по два с каждого регистра). Номера битов, участвующие в генера-

ции Z (как видно из рис. 101), соответствуют значениям 66, 93, 162, 177, 243, 288.

8. АНАЛИЗ СИММЕТРИЧНЫХ АЛГОРИТМОВ ШИФРОВАНИЯ

Современные алгоритмы шифрования разрабатываются таким образом, чтобы аналитик имел как можно меньше шансов отыскать секретный ключ, с помощью которого были зашифрованы данные, даже если ему известен сам алгоритм шифрования и есть в наличии несколько текстов и соответствующих им шифр-текстов. Приступая к задаче анализа, первым делом аналитик определяет тот набор данных, который ему изначально известен для анализа. От этого зависит тот тип криптоанализа, который аналитик сможет использовать. В настоящее время выделяют следующие основные типы криптоанализа.

Криптоанализ на основе только известного шифр-текста. Это наиболее мощная криптоаналитическая атака, так как для ее осуществления криптоаналитику требуется только пассивное прослушивание с целью получения шифр-текстов. При этом имеются минимальные сведения об открытых текстах. В таких условиях один из возможных подходов криптоанализа заключается в простом переборе всех возможных вариантов ключей. Однако если пространство возможных ключей очень велико, этот подход становится нереальным. Поэтому криптоаналитику приходится больше полагаться на анализ самого шифрованного текста, что, как правило, означает выявление его различных статистических особенностей. Криптоаналитик должен иметь некоторые общие предположения о содержимом открытого текста. Например, он может знать, на каком языке был написан зашифрованный открытый текст, или что это исполняемый файл определенной операционной системы, или, что это исходный код программы на каком-либо языке программирования и т.п. Алгоритмы шифрования, которые поддаются атаке такого типа, являются наглядным примером неправильного построения шифров и пригодны лишь для обучения будущих криптоаналитиков.

Криптоанализ на основе известного открытого текста. Этот вид криптоанализа основывается на том, что у

атакующего есть какая-то часть подвергшихся зашифрованию данных. Целью является либо извлечь из этой части известных текстов секретный ключ или хотя бы суметь прочитать неизвестную часть зашифрованных данных. Такой вид криптоанализа до сих пор широко используется, так как трудно запретить противнику предугадывать содержимое открытых текстов (раньше этот метод назывался «Методом возможных слов» (probable word method)). Так, например, в файле с открытым текстом может присутствовать стандартный заголовок или идентификатор. Владея подобной информацией, криптоаналитик может восстановить ключ шифрования на основе логических умозаключений и знания того, каким образом был преобразован известный открытый текст в шифр-текст.

Криптоанализ на основе выбранного открытого текста. В этом случае предполагается, что у противника есть возможность зашифровывать выбранные им открытые тексты. На практике это может быть возможно в случае, когда к противнику в руки попал реализованный алгоритм шифрования с неизвестным секретным ключом или когда есть возможность послать выбранный открытый текст владельцу секретного ключа, а затем получить эти данные в зашифрованном виде при передаче их третьей стороне.

В общем случае, если криптоаналитик имеет возможность по своему усмотрению выбрать сообщение и шифровать его, то при правильном выборе сообщений для шифрования он может вполне оправданно надеяться разгадать ключ.

Криптоанализ на основе выбранного шифр-текста. Этот вид криптоанализа аналогичен предыдущему за тем исключением, что требуется выбрать шифр-тексты для данной схемы расшифрования.

Естественно, что если алгоритм может быть взломан при анализе только шифрованного текста, то он является слабым. Поэтому обычно любой алгоритм шифрования разрабатывается так, чтобы он был устойчив к попыткам взлома с помощью анализа с известным открытым текстом. То есть так, чтобы при известном открытом тексте и соответствующем ему

шифрованном тексте было достаточно трудно, а еще лучше – невозможно, определить секретный ключ шифрования.

Основной характеристикой криптографической стойкости шифра относительно того или иного метода анализа является трудоемкость E(r) этого метода. В качестве меры трудоемкости раскрытия шифров обычно используется количество элементарных операций, необходимых для расшифрования сообщения или определения ключа. Под элементарной операцией понимают операцию, выполняемую на конкретной аппаратуре за один шаг ее работы. Трудоемкость расшифрования определяется объемом и характером информации, доступной криптоаналитику [17].

Алгоритм шифрования называется **безусловно защищенным,** или **абсолютно стойким,** в том случае, если шифр-текст, полученный с помощью данного алгоритма шифрования, не содержит достаточной информации для однозначного восстановления соответствующего открытого текста, при этом объем шифрованного текста не играет никакой роли.

Это означает, что независимо от того, сколько времени потратит противник на расшифровку, ему не удастся расшифровать шифрованный текст просто потому, что в шифрованном тексте нет информации, требуемой для восстановления открытого текста [6]. Среди алгоритмов шифрования абсолютно стойких нет. Таким образом, максимум, чего может ожидать пользователь от того или иного алгоритма шифрования, это выполнение хотя бы одного из двух следующих критериев защищенности.

1. Стоимость взлома шифра превышает стоимость расшифрованной информации.

2. Время, которое требуется для того, чтобы взломать шифр, превышает время, в течение которого информация актуальна.

Алгоритм шифрования называется **защищенным по вычислениям**, если он соответствует обоим вышеуказанным требованиям [6].

Проблема заключается в том, что количественно оценить усилия, необходимые для криптоанализа шифрованного текста, созданного с помощью конкретного данного алгоритма шифрования, очень сложно.

Практически все формы криптоанализа для алгоритмов блочного шифрования используют тот факт, что некоторые характерные особенности структуры открытого текста могут сохраняться при шифровании и проявляться в соответствующих особенностях структур шифрованного текста.

8.1. Метод полного перебора

Если известен алгоритм шифрования и есть хотя бы одна пара открытый – шифрованный текст, то самым естественным способом анализа, который сразу приходит в голову, является последовательное опробование всех возможных вариантов ключа, которые могли быть использованы. Опробование производят до тех пор, пока зашифрование открытого текста на очередном ключе не приведет к получению имеющегося шифрованного сообщения. Такой способ анализа в разных источниках литературы имеет разные названия, например «метод атаки в лоб» [2], или «метод полного перебора» [9], или «метод грубой силы» или или «Brut-force атака» [17]. У этого метода есть одно неоспоримое преимущество: рано или поздно искомый ключ будет найден и для этого будет необходим минимальный набор данных. Быстрота нахождения ключа будет зависеть от длины используемого секретного ключа и от вычислительной мощи, которая есть в наличии у аналитика, а также от доли везения. Ведь может случиться так, что искомый ключ встретится одним из первых. Рассмотрим уравнение относительно ключа $k \in K$ при известной паре (x, y), где $x \in X$ и $y \in Y$:

$$T(x, k) = y. \qquad (11)$$

Основной характеристикой криптографической стойкости шифра относительно того или иного метода анализа является трудоемкость E(r) этого метода [17]. В качестве меры трудоемкости раскрытия шифров обычно используется количество эле-

ментарных операций, необходимых для расшифрования сообщения или определения ключа. Под элементарной операцией понимают операцию, выполняемую на конкретной аппаратуре за один шаг ее работы. Трудоемкость расшифрования определяется объемом и характером информации, доступной криптоаналитику.

Пусть для простоты для любой пары (x, y) существует единственное значение k, удовлетворяющее (11). Упорядочим множество K в соответствии с заданным порядком и будем последовательно проверять ключи из K на предмет равенства в уравнении (11). Если считать проверку одного варианта ключа $k \in K$ в уравнении (11) за одну операцию, то полный перебор ключей потребует |K| операций. Пусть ключ в схеме шифрования выбирается случайно и равновероятно из множества K. То-

гда с вероятностью $\frac{1}{|K|}$ трудоемкость метода полного перебо-

ра равна 1. Это происходит в том случае, когда случайно выбран ключ, расположенный в нашем порядке на первом месте. Поэтому естественно в качестве оценки трудоемкости метода взять математическое ожидание числа шагов в переборе до попадания на использованный ключ. Найдем среднее число шагов в методе полного перебора, когда порядок фиксирован, а выбор ключа случаен и равновероятен.

Пусть случайная величина τ – число опробований включительно до момента обнаружения использованного ключа. При $i = 1, ..., |K|$ случайные величины $\xi_i = 1$, если использованный ключ находится в порядке на месте i и $\xi_i = 0$ в противном случае. Тогда

$$E_\tau = \sum_{i=1}^{|K|} iP(\xi_i = 1). \qquad (12)$$

Если считать, что все ключи расположены в установленном порядке, то процедуру равновероятного выбора ключа можно представлять как равновероятный выбор числа i в последова-

тельности натуральных чисел 1, ..., |K|. Тогда $P(\xi_i = 1) = \frac{1}{|K|}$ для любого i = 1, ..., |K|.

Подставляя полученные значения в (12), получим:

$$E_\tau = \frac{1}{|K|}\sum_{i=1}^{|K|} i = \frac{|K|*(|K|+1)}{|K|*2}.$$

При больших |K| можно приблизительно считать $E\tau \approx \frac{|K|}{2}$ [17].

Вместе с тем нам известно, что одним из важных свойств информации является ее своевременность. Поэтому применение метода полного перебора на практике легко реализуется, но как правило не используется. Так, например, когда разрабатывался алгоритм шифрования DES, длина его фактического секретного ключа была определена в 56 бит. То есть для того, чтобы перебрать все возможные варианты секретных ключей, необходимо было сделать 2^{56} опробований. С помощью имевшихся в то время вычислительных средств это можно было бы сделать за несколько десятков лет! Конечно, с той поры как был разработан алгоритм шифрования DES, в развитии вычислительной техники произошел огромный скачок, и вычислительные мощи возросли в тысячи раз. Сегодня с использованием мощных вычислительных кластеров задача по поиску секретного ключа для алгоритма DES может быть решена за несколько часов. В связи с тем, что вычислительная мощь с каждым днем неумолимо растет, стандарт DES был заменен на новый стандарт AES (Advanced Encryption Standard), где длина секретного ключа возросла до 128 бит. Так или иначе, в криптографии принято время анализа с помощью метода полного перебора считать эталонным. Что это означает? Это значит, что если аналитику удастся провести анализ алгоритма шифрования быстрее, чем это можно сделать с помощью полного перебора, то данный алгоритм шифрования будет считаться уязвимым, в

связи с чем его использовать для шифрования данных будет нецелесообразно.

Задача поиска секретного ключа шифрования методом полного перебора хорошо распараллеливается и легко может быть реализована для многопроцессорных вычислительных систем.

8.2. Метод встречи посередине

Метод «встреча посередине» применим к алгоритмам шифрования, в которых используется два различных ключа K. Это может быть достигнуто в том случае, если секретные подключи появляются с какой-то периодичностью, или, например, если было произведено двойное зашифрование данных, т. е. сначала данные зашифровали на одном ключе K_1, а затем полученный результат шифрования еще раз зашифровали на другом секретном ключе K_2.

Пусть нам известна пара открытый – закрытый текст, зашифрованная подобным образом. В этом случае необходимо произвести зашифрование открытого текста на всех возможных значениях ключа K_1. Параллельно с этим необходимо произвести расшифрование закрытого текста на всех возможных значениях ключа K_2. Та пара ключей (K_1, K_2), для которой результат шифрования открытого текста и результат расшифрования закрытого текста совпадут, и будет являться искомой.

Как видно из объяснений, анализ на основе метода «встреча посередине» может быть распараллелен и реализован с использованием распределенных многопроцессорных вычислений. В качестве примера работы метода можно рассмотреть варианты анализа двойного алгоритма DES или, например, анализа алгоритма ГОСТ 28147-89, в котором один и тот же ключ фактически используется четыре раза.

8.3. Линейный криптоанализ

Метод линейного криптоанализа впервые был предложен в начале 90-х гг. XX в. японским ученым М. Матсуи (Matsui). В своей работе [18] он показал, как можно осуществить атаку на алгоритм шифрования DES, сократив сложность анализа до 2^{47}.

Существенным недостатком метода стала необходимость иметь в наличии большой объем данных, зашифрованных на одном и том же секретном ключе, что делало метод малопригодным для практического применения к вскрытию шифра. Однако, если предположить, что к аналитику в руки попал шифрованный текст, содержащий важные сведения, а также некий черный ящик (устройство или программа), который позволяет выполнить любое число текстов, зашифрованных с помощью известного алгоритма шифрования на секретном ключе, не раскрывая при этом самого ключа, то применение метода линейного криптоанализа становится вполне реальным. Многие алгоритмы шифрования, известные на момент опубликования работы [18], впоследствии были проверены на устойчивость к этому методу и не все из них оказались достаточно стойкими и, как следствие, потребовали доработки.

Знание механизмов работы метода линейного криптоанализа позволяет криптографам еще на этапе проектирования крптоалгоритмов обеспечить стойкость шифров. Вот почему так важно уметь применять известные методы криптоанализа на практике.

Итак, рассмотрим основные понятия, связанные с методом линейного криптоанализа. Любой алгоритм шифрования в самом общем виде можно представить как некоторую функцию E, зависящую от входного сообщения X, секретного ключа K и возвращающую шифрованное сообщение Y:

$$Y = E(X, K). \qquad (13)$$

Зная само преобразование E и входное сообщение X, нельзя однозначно сказать, каким будет выходное сообщение Y. В данном случае нелинейность функции (13) зависит от внутренних механизмов преобразования E и секретного ключа K. М. Матсуи показал, что существует возможность представить функцию шифрования (13) в виде системы уравнений, которые выполняются с некоторой вероятностью p. При этом для успешного проведения анализа вероятность уравнений p должна быть как можно дальше удалена от значения 0,5 (т. е. прибли-

жаться либо к нулю либо к единице). Так как уравнения, получаемые в ходе анализа криптоалгоритма, являются вероятностными, их стали называть линейными статистическими аналогами.

Определение. Линейным статистическим аналогом нелинейной функции шифрования (13) называется величина Q, равная сумме по модулю 2 скалярных произведений входного вектора X, выходного вектора Y и вектора секретного ключа K соответственно с двоичными векторами α, β и γ, имеющими хотя бы одну координату, равную единице:

$$Q = (X,\alpha) \oplus (Y,\beta) \oplus (K,\gamma)$$

в том случае, если вероятность того, что Q=0 отлична от 0,5 ($P(Q=0) \neq 0,5$).

В отличие от дифференциального криптоанализа, в котором большое значение вероятности гарантирует успех атаки, в линейном криптоанализе успех анализа может быть обеспечен как уравнениями с очень большой вероятностью, так и уравнениями с очень маленькой вероятностью. Для того чтобы понять, какое из возможных уравнений лучше всего использовать для анализа, используют понятие отклонения.

Определение. Отклонением линейного статистического аналога называют величину $\eta = |1 - 2p|$, где p – вероятность, с которой выполняется линейный аналог.

Отклонение определяет эффективность линейного статистического аналога. Чем отклонение больше, тем выше вероятность успешного проведения анализа. Фактически отклонение показывает, насколько вероятность статистического аналога отдалена от значения $p = 0,5$.

Для успешного применения метода линейного криптоанализа необходимо решить следующие задачи:

1. Найти максимально эффективные (или близкие к ним) статистические линейные аналоги. При нахождении аналогов обратить внимание на то, что в них должно быть задействовано как можно больше битов искомого секретного ключа K.

2. Получить статистические данные: необходимый объем пар текстов (открытый – закрытый текст), зашифрованных с помощью анализируемого алгоритма на одном и том же секретном ключе.

3. Определить ключ (или некоторые биты ключа) путем анализа статистических данных с помощью линейных аналогов.

Первый шаг анализа заключается в нахождении эффективных статистических аналогов. Для алгоритмов шифрования, в которых все блоки заранее известны, этот шаг можно выполнить единожды, основываясь на анализе линейных свойств всех криптографических элементов шифра. В результате анализа должна быть получена система уравнений, выполняющихся с некоторыми вероятностями. Левая часть уравнений должна содержать в себе сумму битов входного и выходного сообщения, правая часть уравнения – биты секретного ключа. Система уравнений должна быть определенной, т. е. содержать все биты исходного секретного ключа. Данный этап не является вычислительно сложным, однако требует больших знаний, логики работы и внимательности. Он может быть автоматизирован. Однако при этом необходимо помнить, что для каждого определенного алгоритма шифрования система линейных аналогов строится всего один раз и в дальнейшем может быть использована для нахождения разных секретных ключей шифрования, которые используются для шифрования данных с помощью анализируемого шифра.

Если первый шаг анализа является чисто теоретическим и полностью зависит от структуры алгоритма, то второй шаг является исключительно практической частью, которая заключается в анализе известных пар открытый – закрытый текст с помощью полученной ранее системы статистических аналогов. Для этого используется следующий алгоритм.

Алгоритм. Пусть N – число всех открытых текстов и T – число открытых текстов, для которых левая часть линейного статистического аналога равна 0. Рассмотрим два случая.

1. Если $T>N/2$, то в этом случае число открытых текстов, для которых левая часть аналога равна нулю, больше половины,

т. е. в большинстве случаев в левой части аналога появляется значение, равное нулю:

а) если вероятность этого линейного статистического аналога $p > 1/2$, это говорит о том, что в большинстве случаев правая и левая части аналога равны, а значит, левая часть аналога, содержащая биты ключа, равна 0;

б) если вероятность этого линейного статистического аналога $p < 1/2$, это говорит о том, что в большинстве случаев правая и левая части аналога не равны, а значит, левая часть аналога, содержащая биты ключа, равна 1.

2. Если $T<N/2$, то в этом случае число открытых текстов, для которых левая часть аналога равна нулю, меньше половины, т. е. в большинстве случаев в левой части аналога появляется значение, равное единице, то:

а) если вероятность этого линейного статистического аналога $p > 1/2$, это говорит о том, что в большинстве случаев правая и левая части аналога равны, а значит, левая часть аналога, содержащая биты ключа, равна 1;

б) если вероятность этого линейного статистического аналога $p < 1/2$, это говорит о том, что в большинстве случаев правая и левая части аналога не равны, а значит, левая часть аналога, содержащая биты ключа, равна 0.

Пользуясь приведенным выше алгоритмом, можем обобщить вышесказанное для уравнения $X \oplus Y = K$.

Если $T>N/2$, то

$$K = \begin{cases} 0, \text{ если } p > \frac{1}{2}, \\ 1, \text{ если } p < \frac{1}{2}. \end{cases}$$

Если $T<N/2$, то

$$K = \begin{cases} 1, \text{ если } p > \frac{1}{2}, \\ 0, \text{ если } p < \frac{1}{2}. \end{cases}$$

Успех алгоритма возрастает с ростом N и $\Delta = |1 - 2p|$.

Данный алгоритм будет иметь успех при анализе большого числа текстов N. Следовательно, второй шаг анализа является вычислительно сложным. Поэтому для ускорения времени анализа можно и нужно использовать параллельные вычисления.

В результате работы вышеприведенного алгоритма будет получена определенная (а возможно, и переопределенная) система уравнений, отражающая взаимосвязь битов ключа. Третий шаг анализа заключается в решении данной системы, например, методом Гаусса, что позволит получить значения битов секретного ключа шифрования.

Более подробно о линейном криптоанализе различных блочных алгоритмов шифрования можно прочитать в работе [9].

8.4. Дифференциальный криптоанализ

Метод дифференциального криптоанализа впервые был предложен в начале 90-х гг. прошлого века Э. Бихамом и А. Шамиром для анализа алгоритма шифрования DES. Хотя в книге Б. Шнайера [2] упоминается о том, что разработчики алгоритма DES знали о возможности такого анализа еще во время разработки алгоритма в 70-х гг. XX в., широкая общественность узнала о дифференциальном криптоанализе именно из работ [19, 20]. Метод ДК оказался первым методом, позволяющим взломать DES при оценке сложности задач менее 2^{55}. Согласно [19], с помощью данного метода можно провести криптоанализ DES при усилиях порядка 2^{37}, но при наличии 2^{47} вариантов избранного открытого текста. Хотя 2^{47}, очевидно, значительно меньше, чем 2^{55}, необходимость при этом иметь 2^{47} вариантов избранного открытого текста превращает данный вариант схемы криптоанализа в чисто теоретическое упражнение [6]. Это связано с тем, что метод ДК был известен в момент разработки DES, но засекречен по очевидным соображениям, что подтверждается публичными заявлениями самих разработчиков [2]. В работе [20] показано, что если поменять порядок следования блоков замены в алгоритме шифрования DES или использовать другие наборы таблиц подстановок и перестановок, то алгоритм становится сразу намного слабее и может быть взломан менее

чем за половину времени, требуемого для анализа алгоритма DES с помощью полного перебора. Это показывает значимость знания возможных путей анализа разрабатываемого алгоритма.

С помощью метода дифференциального криптоанализа (differential cryptanalysis), предложенного Э. Бихамом и А. Шамиром [19, 20], сложность анализа сократилась до 2^{37}. Однако при этом для проведения анализа необходимо было иметь 2^{37} особым образом подобранных текстов, зашифрованных на одном и том же секретном ключе. Несмотря на накладываемые ограничения в использовании новых предложенных методов анализа – это был прорыв! Дальнейшее развитие этого метода показало возможность его применения к целому классу различных видов шифров, позволило выявить слабые места многих используемых и разрабатываемых алгоритмов шифрования. Сегодня этот метод, а также некоторые его производные, такие как метод линейно-дифференциальный, метод невозможных дифференциалов, метод бумеранга, широко используются для оценки стойкости вновь создаваемых шифров. Именно поэтому специалисту по защите информации необходимо иметь представление о механизмах анализа шифров с использованием современных методов криптоанализа.

Само название дифференциальный криптоанализ происходит от английского слова difference, т. е. разность. Именно поэтому в отечественной литературе этот вид анализа еще иногда называют разностным методом. Исходя из названия, можно понять, что при рассмотрении возможности анализа некоторого блочного алгоритма шифрования ученым пришло в голову использовать не отдельные тексты, а пары текстов. Понятно, что два текста будут иметь различия в некоторых позициях. Для того чтобы определить это различие, достаточно пару текстов сложить между собой по модулю 2. Результат такого сложения даст на выходе значение 0 в тех позициях, в которых исходные тексты были равны между собой, и соответственно значение 1 в тех позициях, в которых исходные тексты отличались. Например, рассмотрим два 4-битовых сообщения: X = 0011 и X' = 1010. В результате сложения текстов X и X' была получена разность ΔX = 1001, полученное значение ΔX принято называть

дифференциалом, или разностью. В дифференциальном криптоанализе значение разности (дифференциала) принято обозначать символом Δ. Разность, полученная в результате сложения текстов X и X', показывает, что во второй и третьей позициях исходные сообщения X и X' были равны, а в первой и четвертой отличались друг от друга.

Здесь целесообразно будет ввести несколько определений, характерных для метода дифференциального криптоанализа, которыми мы будем оперировать в дальнейшем. Для большей наглядности основные значения пояснены с помощью рис. 102.

Разность (дифференциал) – результат поразрядного сложения по модулю 2 (операция XOR) двух отдельно шифруемых на одном и том же секретном ключе текстов, находящихся в одной и той же позиции рассматриваемого алгоритма шифрования.

Входная разность (входной дифференциал) – разность, поступающая на вход рассматриваемого криптографического преобразования.

Выходная разность (выходной дифференциал) – разность, образованная на выходе рассматриваемого преобразования.

Характеристика – это пара дифференциалов, один из которых образован входными значениями некоторого преобразования, а второй – выходными значениями этого же преобразования. Дифференциал на входе преобразования также часто называют ***входной разностью***, а дифференциал на выходе преобразования – ***выходной разностью***.

Раундовая характеристика – это пара дифференциалов, один из которых образован входными значениями раунда шифрования, а второй – выходными значениями того же раунда.

Вероятность характеристики – это вероятность, с которой выходная разность характеристики будет получена для заданной входной разности характеристики.

Правильная пара текстов – две пары значений открытое сообщение – шифрованное сообщение, для которых разность входных сообщений равна входной разности характеристики и

разность шифрованных сообщений равна выходной разности характеристики.

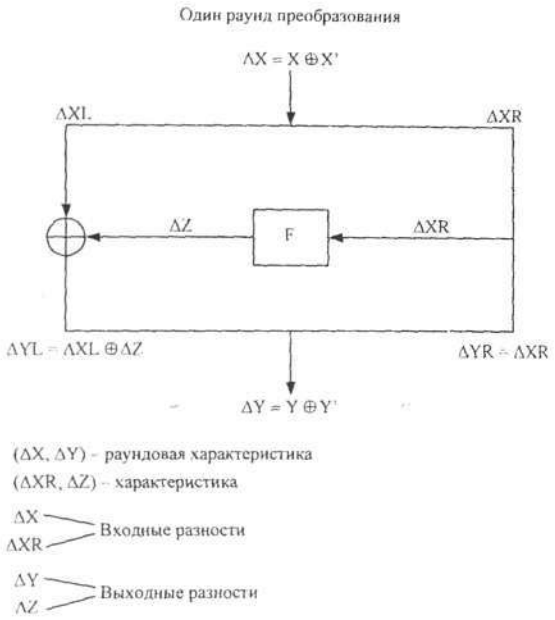


Рис. 102. Основные значения дифференциального критоанализа

В общем виде дифференциальный анализ блочных алгоритмов шифрования сводится к следующим основным пунктам.

1. Нахождение для алгоритма шифрования характеристик, обладающих максимальными характеристиками. Поиск характеристик ведется на основе дифференциальных свойств нелинейных криптографических примитивов, входящих в состав алгоритма шифрования.

2. Поиск правильных пар текстов с использованием найденных характеристик.

3. Анализ правильных пар текстов и накопление статистики о возможных значениях секретного ключа шифрования.

Первый пункт, заключающийся в поиске лучших характеристик, для большинства алгоритмов выполняется единожды и

является теоретической задачей. Значения характеристик полностью зависят от структуры алгоритма шифрования и используемых криптографических примитивов. Иначе дело обстоит лишь с теми алгоритмами, которые обладают нефиксированными элементами. К таким алгоритмам можно, например, отнести алгоритм шифрования ГОСТ 28147-89, у которого S-блоки замены могут выбираться произвольным образом. Для таких алгоритмов поиск характеристик необходимо каждый раз начинать сначала, основываясь на дифференциальных свойствах выбранных S-блоков. Для автоматизации процесса анализа можно разработать алгоритм поиска лучших характеристик, основываясь на алгоритмах поиска по дереву. Для таких алгоритмов можно использовать параллельные модели для ускорения поиска характеристик.

Второй шаг анализа является вычислительно стойкой задачей для любого алгоритма шифрования, при этом не важно, обладает он фиксированными или нефиксированными элементами. Анализ заключается в опробовании большого числа пар текстов с целью определения, являются ли они правильной парой текстов, т. е. той парой текстов, которую в дальнейшем можно использовать для анализа с целью поиска секретного ключа шифрования. Данный шаг может и должен быть легко представим в виде параллельных вычислений для сокращения времени анализа.

Последний шаг легко реализуем, требует гораздо меньше вычислений в сравнении со вторым шагом. Он может быть реализован как отдельно в виде последовательного алгоритма, так и быть включенным в состав параллельных алгоритмов по поиску правильных пар текстов. В последнем случае при нахождении правильной пары текстов сразу можно провести ее анализ по накоплению статистики о возможном значении секретного ключа.

Более подробно о дифференциальном криптоанализе различных блочных алгоритмов шифрования, а также о различных производных данного метода можно прочитать в работах [7, 9, 19, 20].

8.5. Алгебраический анализ

Сущность алгебраических методов анализа заключается в получении уравнений, описывающих нелинейные преобразования замены S-блоков, с последующим решением найденных систем уравнений и получением ключа шифрования. Данный метод криптоанализа относится к атакам с известным открытым текстом, поэтому для успешного анализа достаточно иметь одну пару открытый текст/шифр-текст. Алгебраические методы криптоанализа состоят из следующих этапов:

– составление системы уравнений, описывающей преобразования в нелинейных криптографических примитивах анализируемого шифра (чаще всего для симметричных алгоритмов шифрования такими нелинейными компонентами являются S-блоки замены);

– решение полученной системы уравнений.

Рассмотрим подробнее первый этап алгебраического криптоанализа. Для шифров, подобных Rijndael, при составлении уравнений используется таблица замены S-блоков. Ограничимся рассмотрением одночленов, состоящих из произведения двух переменных. Тогда уравнения, описывающие работу S-блоков, имеют вид [21]:

$$\sum \alpha_{ij} x_i x_j + \sum \beta_{ij} y_i y_j + \sum \gamma_{ij} x_i y_j + \sum \delta_i x_i + \sum \varepsilon_i y_i + \eta = 0,$$

где $x_i x_j$ – комбинация входных битов S-блока;

$y_i y_j$ – комбинация выходных битов S-блока;

$x_i y_j$ – комбинация входных и выходных битов;

x_i и y_i – соответственно входные и выходные биты S-блока;

η – коэффициент, принимающий значения 0 или 1.

При получении уравнений нужно рассмотреть все возможные комбинации данных одночленов. В случае, когда число битов на входе S-блоков равно s, получаем, что число одночленов, встречающихся в системе, вычисляется по формуле $t=\frac{2s}{2}+2s+1$ и включает в себя входные и выходные значения S-блока (2s),

все их возможные произведения $\binom{2s}{2}$ и коэффициент η. Число всех возможных комбинаций одночленов составляет 2^t.

Для произвольного блока замены число линейно независимых уравнений $r \geq t - 2^s$.

Для проверки всех полученных комбинаций на соответствие заданному S-блоку требуется составить таблицу истинности на основании замен, выполняемых в исследуемом S-блоке (табл. 39).

Для проверки комбинаций на соответствие таблице истинности следует осуществить построковую подстановку значений одночленов из таблицы и выполнить операцию сложения по модулю 2. Таким образом, для каждой комбинации выполняется подстановка и сложение для всех возможных входных значений S-блока (2^s раз). Результаты суммирования сравниваются с нулем. Если для всех строк таблицы истинности равенство оказывается верным, то уравнение, заданное данной комбинацией одночленов, удовлетворяет таблице замены исследуемого S-блока, и его следует отобрать для составления искомой системы. Далее необходимо провести анализ уравнений и выбрать для формирования системы линейно независимые уравнения, содержащие минимальное число нелинейных элементов.

Таблица 39

Общий вид таблицы истинности для S-блока

	Входные значения S-блока			Выходные значения S-блока			Все сочетания входных и выходных значений S-блока										
Все возможные входные значения S-блока (от 0 до 2^s)	x_n	...	x_1	y_n	...	y_1	x_1x_{n-1}	...	x_2x_1	y_1y_{n-1}	...	y_2y_1	x_1y_n	...	x_1y_1	η
	0	...	0	1	...	1										1
	...															
	1	...	1	0	...	1										1

Второй этап алгебраического криптоанализа заключается в решении системы. В криптоанализе разработаны различные подходы к решению нелинейных систем булевых уравнений. Наиболее эффективными, как показывает практика криптоанализа, являются методы, использующие линеаризацию исходной системы.

XL-метод (eXtended Linearization) предложен Nicolas Courtois, Alexander Klimov, Jacques Patarin и Adi Shamir в работе [22].

Пусть имеется нелинейная система, содержащая m уравнений и 2s переменных. XL-метод базируется на умножении каждого уравнения 1…m на произведения переменных степени, меньшей или равной D–2. Рассмотрим вычисление параметра D алгоритма XL атаки. При умножении исходных уравнений системы на одночлены степени $\leq$(D–2) получаем примерно $R \approx \binom{2s}{\mathrm{D}-2} m$ новых уравнений. Общее число одночленов, встречающихся в этих уравнениях, составляет $T = \binom{2s}{D}$. Так как система будет решаться способом линеаризации, то есть путем замены всех нелинейных одночленов на новые переменные, необходимо чтобы число уравнений было больше числа одночленов $R = \binom{2s}{\mathrm{D}-2} m \geq \binom{2s}{D} = T$. Отсюда получаем, что $m \geq \binom{2s}{D} / \binom{2s}{\mathrm{D}-2} \approx (2s)^2 / D^2$. Следовательно, $D \approx \frac{2s}{\sqrt{m}}$. При этом должно выполняться условие D>2, иначе не будет получено новых уравнений, так как степень отобранных для умножения уравнений одночленов, определяемая разностью D–2, будет равна нулю.

Алгоритм XL метода состоит из двух шагов.

1. Multiply – умножение каждого уравнения исходной системы на произведение переменных в степени $\leq$ D–2.

2. Linearize – замена каждого одночлена в степени $\leq$ D на новую переменную и применение метода исключения Гаусса.

Сложность анализа заключается в построении системы всех возможных линейных уравнений и последующего ее решения. Для ускорения процесса анализа построение уравнений для системы можно производить параллельно. Также переопределенную систему систему многих уравнений целесообразно решать с использованием параллельных вычислений с последующим объединением результата.

Более подробно об алгебраическом криптоанализе различных блочных алгоритмов шифрования можно прочитать в работах [21-25].

8.6. Анализ стандарта AES

Атака типа «Квадрат»

С появлением нового стандарта шифрования AES, в основе которого лежит алгоритм шифрования Rijndael, стал рассматриваться новый принцип построения блочных шифров. По принципу построения и обработки данных стали выделять новую архитектуру под названием «Квадрат». С появлением новой архитектуры в алгоритмах шифрования возникла потребность в разработке новых подходов в анализе подобных криптосистем. Рассмотрим атаку типа «Квадрат» на примере упрощенного алгоритма Rijndael. Для этого к анализу возьмем четырехраундовый алгоритм шифрования Rijndael.

Возьмем 256 входных сообщений, имеющих одинаковые значения во всех байтах кроме одного. Так как операция SubBytes является простой операцией замены одного значения на другое, то после этой операции первого раунда данные все еще будут иметь различия в том единственном байте.

Пусть у нас есть два входа преобразования MixColumn (a, b, c, d) и (a', b, c, d), тогда соответствующие им выходные значения будут иметь разность (02_x•(a – a'), 01_x•(a – a'), 01_x•(a – a'), 03_x•(a – a')), где все операции производятся в поле $GF(2^8)$. Следовательно, выходные значения преобразования MixColumn будут иметь отличия во всех четырех байтах. А значит, в нашем случае все четыре байта столбца могут принять любое значение из 256 возможных. Как мы уже выяснили ранее, после преобразования третьего раунда MixColumn выходные данные при

сложении друг с другом по модулю 2 в результате дадут ноль. А значит, и перед операцией SubBytes четвертого раунда шифрования это свойство сохранится.

Таким образом, анализ алгоритма шифрования Rijndael может быть проведен с помощью выполнения следующих действий.

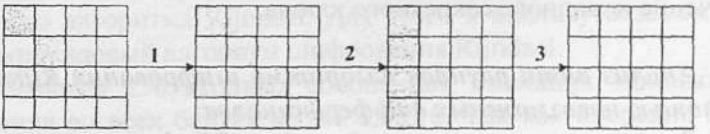
1. Необходимо выбрать 256 открытых текстов, которые будут иметь различия только в первом байте.
2. Получить соответствующие им шифр-тексты, зашифровав их с помощью четырех раундов шифрования Rijndael на секретном ключе.
3. Предположить значение (т. е. взять одно из возможных) первого байта подключа четвертого раунда шифрования.
4. Используя 256 известных шифр-текстов, получить 256 значений первого байта на входе преобразования SubBytes четвертого раунда.
5. Если сумма полученных 256 значений по модулю 2 равна нулю, то предположенное значение байта подключа верно.
6. Используя этот же принцип, найти остальные байты подключа четвертого раунда шифрования.
7. Зная значение подключа четвертого раунда, извлечь значение исходного секретного ключа.

Анализ пяти раундов алгоритма шифрования Rijndael с помощью невозможных дифференциалов

Рассмотрим алгоритм шифрования Rijndael, состоящий из 4-х циклов. Если пара открытых текстов имеет различие только лишь в одном байте, то соответствующие им шифр-тексты не могут иметь одинаковые значения в следующих комбинациях байтов: (1, 6, 11, 16), (2, 7, 12, 13), (3, 8, 9, 14) или (4, 5, 10, 15). Это происходит вследствие того, что перед преобразованием MixColumn в первом раунде различие имеется только в одном байте, после преобразования – во всем столбце, а после преобразования MixColumn второго раунда – во всех 16 байтах. С другой стороны, если шифр-тексты имеют одинаковые значения в одной из запрещенных комбинаций байтов, то после преобразования MixColumn третьего раунда данные имеют одина-

ковые значения в одном из столбцов, а это значит, что и до преобразования MixColumn третьего раунда данные также имеют одинаковые значения в этом столбце. Следовательно, после преобразования MixColumn второго раунда в данных осталось четыре байта, равных между собой. Это противоречит тому, что мы выяснили ранее: после преобразования MixColumn данные должны иметь различия во всех байтах. Заметим, что вероятность возникновения такого противоречия при рассмотрении случайной пары текстов равна 2^{-30} [26].

Таким образом, анализ пяти раундов алгоритма шифрования Rijndael основан на исключении неправильных подключей первого раунда, при использовании которых в последних четырех раундах появляется значение невозможной разности. Нам необходимо рассмотреть такие пары текстов, которые на входе второго раунда шифрования будут иметь различие только в первом байте. Это возможно в том случае, когда на вход алгоритма шифрования поступает пара текстов, имеющая различие в первом, шестом, одиннадцатом и шестнадцатом байтах (рис. 100). На рис. 103 серым цветом обозначены байты, в которых тексты не имеют сходства.



1 – Операция сложения с раундовым ключом AddRoundKey()

2 – Операция циклического сдвига ShiftRows()

3 – Операция преобразования столбцов MixColumns()

Рис. 103. Анализ алгоритма шифрования Rijndael

Зная это, можно провести анализ данного алгоритма шифрования по следующей схеме.

1. Выбрать 2^{63} пар открытых текстов, таких, что разность в первом, шестом, одиннадцатом и шестнадцатом байтах у них не будет равна нулю.
2. Зашифровать их на одном и том же секретном ключе.

3. Выбрать те пары текстов, у которых разность шифр-текстов имеет ненулевые значения в одной из невозможных комбинаций байтов. Таких пар будет примерно $2^{63} * 2^{-30} \approx 2^{33}$.

4. Для каждой такой пары открытых текстов (P, P*) выполнить следующие действия:

а) предположить значение первого, шестого, одиннадцатого и шестнадцатого байтов первого подключа K^0;

б) вычислить, чему будут равны первый, пятый, девятый и тринадцатый байты выхода первого раунда для шифр-текстов P и P*, зашифровав их с помощью выбранных байтов подключа K^0;

в) если разность первых вычисленных байтов отлична от нуля, а разность остальных вычисленных байтов равна нулю, то байты подключа K^0 выбраны неверно;

г) продолжать выполнять действия пп. а, б и в до тех пор, пока не будет найдено верное значение байтов первого подключа.

Остальные байты первого подключа могут быть найдены с использованием аналогичной схемы, однако рассматривать в этом случае надо не первый столбец второго раунда, а все остальные.

Обе вышеприведенные атаки на алгоритм Rijndael легко могут быть реализованы с использованием распределенных многопроцессорных вычислений. Особенно это актуально для метода анализа на основе невозможных дифференциалов, где для анализа необходимо обработать довольно большой массив данных. Более подробную информацию об анализе алгоритма AES можно найти в работах [7, 9, 26].

8.7. Слайдовая атака

С ростом скорости современных компьютеров, скоростные алгоритмы шифрования используют все больше и больше раундов, признавая все существующие криптоаналитические технологии бесполезными. Это, главным образом, происходит из-за того, что такие популярные методы, как линейный и дифференциальный криптоанализ, являются статистическими атаками, превосходными при статистических непостоянствах.

Однако, когда алгоритм шифрования имеет большое количество раундов, каждый добавленный раунд требует экспоненциального роста усилий атакующего.

Стремление к большому числу раундов можно наглядно увидеть, рассмотрев претендентов на конкурс AES. Несмотря на то, что одним из основных критериев для претендентов была скорость, некоторые представленные кандидаты (при этом не самые медленные) имели действительно большое число раундов: RC6(20), MARS(32), SERPENT(32), CAST(48). Это является следствием того, что после некоторого большого числа раундов даже относительно слабый шифр становится стойким. Например, алгоритм шифрования DES, уже взлом шестнадцати раундов представляет трудную задачу, не говоря о 32 и 48 раундах (двойном и тройном алгоритме DES). Таким образом, для криптоаналитика становится естественным поиск новых методов анализа, не зависящих от числа раундов в алгоритме шифрования.

В этом подразделе мы рассмотрим новый метод криптоанализа, не зависящий от числа раундов в алгоритме шифрования, который называется "слайдовой атакой" или "скользящей атакой" (Slide Attacks), предложенный в 1999 г. А. Бирюковым и Д. Вагнером [27]. Этот метод применим ко всем алгоритмам блочного шифрования.

В то время как два других метода криптоанализа, таких как линейный и дифференциальный, концентрируются главным образом на распространенных свойствах техники шифрования, слайдовая атака использует степень самоподобия, что является принципиальным отличием. Под самоподобием понимается использование одной и той же криптографической F-функции, зависящей от одного и того же подключа, в каждом раунде шифрования. В зависимости от структуры алгоритма шифрования слайдовая атака может использовать как слабость процедуры формирования подключей, так и более общие структурные свойства шифра. Самый простой вид этой атаки обычно легко пресечь, избавившись от самоподобия в алгоритме шифрования. Более сложные варианты этой атаки имеют более сложный анализ, и против них гораздо труднее защититься.

Самый простой вариант слайдовой атаки рассчитан на анализ алгоритмов шифрования, состоящих из r раундов, каждый из которых содержит в себе F-функцию, зависящую от одного и того же значения ключа K. Такой тип алгоритмов шифрования называется гомогенным. К гомогенным также относятся алгоритмы, в которых функция формирования подключа периодична, т. е. один и тот же подключ извлекается через равное количество раундов. Говоря математическим языком, $F_i = F_j$ для всех $i \equiv j \bmod p$, где p является периодом. В самом простом случае p=1, и в каждом раунде используется один и тот же подключ.

При рассмотрении слайдовых атак, с целью упрощения мы будем рассматривать алгоритмы шифрования, в которых сложение шифруемых данных с подключом происходит непосредственно перед F-функцией. Так, если мы рассматриваем алгоритм шифрования, построенный по схеме Фейстеля, на вход которого поступает n-битное сообщение, то длина подключа будет составлять n/2 битов.

На рис. 104 показан процесс зашифрования n-битового открытого текста X_0, в результате которого получается шифртекст X_r. Здесь X_j обозначает промежуточное значение данных после j-го раунда зашифрования, так что

$$X_j = F_j(X_{j-1}, k_j),$$

где j = 1, 2, 3, ..., r. В дальнейшем мы иногда будем опускать значение k в обозначении F-функций и будем писать F(x) или $F_i(x)$ вместо F(x,k) или $F_i(x,k)$.

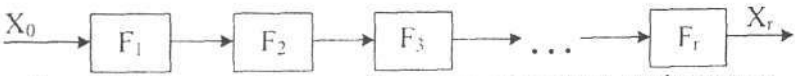


Рис. 104. Схема обычного блочного алгоритма шифрования

Функция F называется слабой в том случае, если при известных двух равенствах F(x1,k) = y1 и F(x2,k) = y2 ключ k легко определяется. На рис. 105 показано, как может быть применена слайдовая атака к алгоритмам шифрования такого типа.

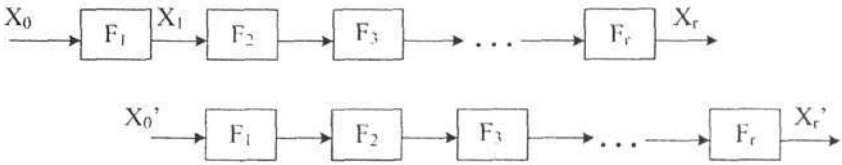


Рис. 105. Схема обычной слайдовой атаки

Идея заключается в том, что можно сопоставить один процесс зашифрования с другим таким образом, что один из процессов будет «отставать» от другого на один раунд.

Пусть X_0 и X_0' обозначают исходные открытые тексты, $X_j = F_j(X_{j-1})$ и $X_j' = F_j(X_{j-1}')$, где j = 1, 2, 3, …, r. Идея заключается в том, что если мы имеем такую пару значений, что $X_1 = X_0'$, то мы также будем иметь соответствующую им пару значений, такую что $X_r = X_{r-1}'$. Предположим, что $X_j = X_{j-1}'$, тогда мы можем сказать, что $X_{j+1} = F(X_j) = F(X_{j-1}') = X_j'$. Пара открытых текстов и соответствующих им шифр-текстов (P,C), (P',C') называется слайдовой парой в том случае, если F(P) = P' и F(C) = C'.

Слайдовая атака проходит следующим образом. Мы получаем $2^{n/2}$ пар открытый – закрытый текст (P_i, C_i), и ищем среди них слайдовые пары. Согласно парадоксу дней рождений, мы ожидаем найти хотя бы одну пару индексов i, i', такую, что $F(P_i) = P_i'$ и $F(C_i) = C_i'$ одновременно выполняются для некоторого ключа. После того как слайдовая пара найдена, мы можем найти некоторые биты подключа. В том случае, если раундовая функция слабая, мы можем найти весь подключ данного раунда. Для нахождения оставшихся битов секретного ключа необходимо определить следующую слайдовую пару, и с ее помощью провести анализ. Таким образом, достаточно найти всего несколько слайдовых пар для полного определения битов секретного ключа, что и является задачей, стоящей перед криптоаналитиком.

В случае, когда речь идет об алгоритмах шифрования, построенных по схеме Фейстеля, раундовая функция F((l,r), k) = ((l⊕f(r),r), k) модифицирует только половину входного сообще-

ния. Таким образом, условие F(x) = x' можно легко проверить с помощью сравнения левой части сообщения x и правой части сообщения x'. Это условие позволяет нам снизить сложность атаки на основе известных открытых текстов до $2^{n/2}$ известных текстов. У нас есть n-битовое условие нахождения потенциальной слайдовой пары: если (P_i, C_i) образует слайдовую пару вместе с (P_i', C_i'), то тогда $F(P_i) = P_j'$ и $F(C_i) = C_j'$. Для нахождения слайдовой пары для алгоритмов шифрования, построенных по схеме Фейстеля, необходимо известные тексты (P_i, C_i) занести в таблицу, после чего для каждого j найти такой текст, чтобы правые половины P_i и C_j' были равны левым половинам P_j' и C_i.

Если не все биты подключа будут найдены с помощью определенной слайдовой пары, то можно будет использовать другие слайдовые пары для определения оставшихся битов.

Для алгоритмов шифрования, построенных по схеме Фейстеля, сложность анализа может быть снижена до $2^{n/4}$ текстов в том случае, если существует возможность использовать выбранные открытые тексты. Для этого необходимо выбрать произвольным образом n/2 битовое значение x. После этого надо подобрать массив из $2^{n/4}$ открытых текстов $P_i = (x, y_i)$, которые будут отличаться только случайно выбранной правой частью, и массив из $2^{n/4}$ текстов $P_j' = (y_j', x)$, которые будут отличаться только случайно выбранной левой частью. Таким образом, у нас появится $2^{n/2}$ пар открытых текстов, и мы надеемся найти среди них хотя бы одну правильную пару.

В любом из вышерассмотренных случаев поиск слайдовых пар является вычислительно сложной задачей, для решения которой целесообразно использовать параллельные алгоритмы.

8.8. Парадокс дней рождений и его роль в задачах криптоанализа

Это очень важный вероятностный парадокс с неперечислимым количеством применений в современной криптографии: от алгоритмов блочного шифрования до систем с открытым ключом.

Предпосылкой возникновения «парадокса дней рождений» явился вопрос: как много учеников должно собраться в одном

классе, чтобы как минимум двое из них имели день рождения в один и тот же день? С помощью простых вычислений можно выяснить, что если в классе будет находиться 23 ученика, то вероятность того, что у двух из них день рождения в один день, будет больше, чем ½.

Итак, пронумеруем учеников в классе от 1 до k и обозначим день рождения i-го ученика как d_i (которое может принимать значение от 1 до n, а n=365). Учитывая, что дни рождения не зависят друг от друга, получаем, что вероятность того, что ученики i и j имеют одинаковый день рождения, равна

$$\sum_{d=1}^{n} P(d_i = d) \bullet (d_j = d) = n \bullet \frac{1}{n^2} = \frac{1}{n}$$

Таким образом, вероятность того, что у учеников i и j день рождения в один день равна вероятности того, что один из них родился в определенный день года. Теперь найдем вероятность того, что как минимум двое из k учеников родились в один день. Вероятность того, что все k учеников имеют разные дни рождения, равна

$$P_{\text{разные дни рождения}} = 1 \bullet \left(1 - \frac{1}{\text{n}}\right) \bullet \left(1 - \frac{2}{\text{n}}\right) \cdots \left(1 - \frac{k-1}{\text{n}}\right) \qquad (14)$$

Ясно, что для двух человек – обозначим их как 1 и 2 – вероятность того, что они родились в один день, равна $\frac{1}{n}$. А значит, вероятность того, что у них разные дни рождения, равна $1 - \frac{1}{n}$. Так как известна вероятность того, что двое имеют разные дни рождения, рассматривая трех учеников, скажем 1, 2 и 3, можно определить вероятность того, что ученик 3 родился с учениками 1 и 2 в разные дни. Эта вероятность равна $1 - \frac{2}{n}$. Таким образом, увеличивая число учеников до k, получим, что

вероятность того, что ученик k не рожден в один день с учениками 1, 2, . . ., k–1, равна $1-\frac{k-1}{n}$.

В том случае, когда вероятность того, что все k учеников родились в разные дни ниже ½, получается, что вероятность того, что как минимум двое из них родились в один день больше или равна ½. Пользуясь неравенством $(1-n) \le e^{-n}$, можем заменить в формуле (14) значение (1–n) на e^{-n}, тогда получаем:

$$P_{\text{разные дни рождения}} \le e^{-1/n} e^{-2/n} \ldots e^{-(k-1)/n} = e^{-\frac{k(k-1)}{2n}}.$$

Значит, вероятность того, что, по крайней мере, у двух учеников дни рождения будут совпадать, равна

$$P_{\text{совпадающие дни рождения}} = 1 - e^{-\frac{k(k-1)}{2n}}.$$

Решая неравенство

$$1 - e^{-\frac{k(k-1)}{2n}} = \frac{1}{2},$$

получаем

$$k \ge \frac{1}{2} + \sqrt{\frac{1}{4} + 2n \ln 2}.$$

Если n достаточно большое, то можно записать:

$$k \approx \sqrt{2n \ln 2}.$$

Если предположить, что n=365, тогда k=22,54, что близко к правильному значению 23.

Применение «парадокса дней рождений» на практике очень распространено. Предположим, что для атаки на алгоритм шифрования, оперирующий 64-битными блоками данных, противник должен получить две пары открытый – закрытый текст, которые отличаются только младшим значащим битом (типичная задача для дифференциального криптоанализа). Согласно «парадоксу дней рождений», только массив, состоящий примерно из 2^{32} открытых текстов, будет содержать требуемые пары с высокой вероятностью. Рассмотрим другой пример для одного раунда 64-битного шифра Фейстеля. Предположим, что в шифре используется произвольная функция F. Злоумышленник может захотеть узнать, как много открытых текстов ему нужно получить для того, чтобы на выходе встретилось два одинаковых шифр-текста F-функции. Согласно «парадоксу дней рождений», можно определить, что в этом случае требуется только 2^{16} текстов.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Перечислите и охарактеризуйте основные операции, используемые в современных шифрах.
2. Охарактеризуйте симметричные системы шифрования.
3. Опишите работу алгоритма блочного шифрования DES.
4. Чем отличается шифрование и дешифрование для алгоритма DES?
5. Для каких целей введен блок расширения E в реализацию функции F алгоритма DES и как он работает?
6. Каким образом связаны перестановки IP и IP^{-1} в алгоритме шифрования DES? Поясните на примере двух – трех элементов.
7. Каким образом происходит преобразование 6-битных элементов векторов в таблицах замены функции F алгоритма DES в 4-битные?
8. Поясните схемы выработки раундовых подключей в алгоритме DES.
9. Опишите схему программно-аппаратной реализации режима простой замены для алгоритма ГОСТ 28147-89.
10. Опишите схему программно-аппаратной реализации режима гаммирования для алгоритма ГОСТ 28147-89.
11. Опишите схему программно-аппаратной реализации режима гаммирования с обратной связью для алгоритма ГОСТ 28147-89.
12. Опишите схему выработки имитовставки для алгоритма ГОСТ 28147-89.
13. Опишите принцип использования секретного ключа для шифрования данных с помощью алгоритма ГОСТ 28147-89.
14. Есть ли в функции F алгоритма ГОСТ 28147-89 перестановки, если есть, как они реализованы?
15. Какой блок в режиме гаммирования ГОСТ 28147-89 зашифровывает открытый текст?
16. Каким образом происходит преобразование 4-битных входных элементов векторов в блоках замены функции F ГОСТ 28147-89 в 4-битные выходные элементы векторов.

17. Поясните кратко, каким образом вырабатывается гамма шифра $\Gamma_ш^{(i)}$ в алгоритме ГОСТ 28147089 для шифрования i-го блока открытого текста или расшифрования шифр-текста ($T_0^{(i)}$ и $T_ш^{(i)}$).

18. Опишите работу алгоритма шифрования Rijndael.

19. Приведите основные характеристики для стандарта AES. Чем стандарт AES отличается от алгоритма Rijndael?

20. Опишите схему выработки раундовых подключей для стандарта AES.

21. Опишите работу алгоритма расшифрования для стандарта AES.

22. Опишите новый шифр Кузнечик (зашифрование и расшифрование).

23. Как вырабатываются раундовые подключи для алгоритма шифрования Кузнечик?

24. Опишите принцип работы преобразования L для алгоритма Кузнечик.

25. В чем заключается слабость схемы двойного шифрования DES?

26. Опишите, как применяется тройное шифрование DES с двумя ключами.

27. Опишите работу режима сцепления блоков CBC.

28. Опишите работу режима обратной связи по выходу OFB.

29. Опишите работу режима обратной связи по шифртексту CFB.

30. Сколько режимов работы предусмотрено для нового отечественного алгоритма шифрования Кузнечик?

31. Охарактеризуйте поточные шифры, укажите их основные свойства.

32. Что представляет собой регистр сдвига с обратной линейной связью?

33. В каком случае период вырабатываемой псевдослучайной последовательности с помощью РСЛОС будет максимальным?

34. Опишите основные этапы работы поточного шифра A5/1.

35. Как работает режим stop-and-go для поточного шифра A5/1?

36. Что представляет собой легковесная криптография и где она применяется?

37. Опишите работу алгоритма Present.

38. Как выполняется шифрование данных с помощью алгоритма Clefia?

39. По какому принципу вырабатываются раундовые подключи для алгоритма шифрования Clefia?

40. Опишите работу шифра Trivium?

41. На какие основные виды можно разделить существующие криптографические атаки?

42. В чем заключается атака «в лоб» или атака полного перебора? Чему равна трудоемкость атаки?

43. В чем заключается суть метода анализа «встреча посередине»?

44. К какому виду криптоатак можно отнести дифференциальный криптоанализ? Почему?

45. В чем заключается дифференциальный криптоанализ? Перечислите основные этапы и приемы.

46. К какому виду криптоатак можно отнести линейный криптоанализ?

47. В чем заключается линейный криптоанализ? Назовите основные этапы и приемы.

48. В чем заключается смысл слайдовой атаки?

49. В чем заключается «парадокс дней рождений» и какова его роль в задачах криптоанализа?

ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

1. Зашифруйте с помощью учебного алгоритма шифрования

№	X	K	Ответ
1	43827	9375679	45732
2	56241	8354877	24402
3	18354	3150189	65518
4	215	13745678	32669
5	3784	3956874	56131
6	53112	5138975	18718
7	55555	10185130	16496
8	10378	11252511	39783
9	31825	5178345	63998
10	812	7845832	29379
11	41576	8123445	4345
12	31567	569784	17307
13	65112	3150192	38911
14	382	7345679	48769
15	25386	8345678	54000
16	7536	12053864	46891
17	8254	1073895	50822
18	13112	6783759	32987
19	44444	7814580	53190
20	27356	3150012	10353

2. Зашифруйте с помощью алгоритма шифрования S-DES

№	X	K	Ответ
1	123	568	26
2	26	378	207
3	159	1003	123
4	46	379	90
5	179	394	3
6	215	735	41
7	27	836	107
8	172	937	92
9	48	236	62
10	84	254	242
11	138	396	180
12	94	390	132
13	29	1000	171
14	250	432	13
15	67	463	234
16	167	542	93
17	94	867	54
18	103	354	43
19	30	365	129
20	96	763	65

3. Зашифруйте с помощью алгоритма шифрования S-AES

№	X	K	Ответ
1	7 5 4 5	8 6 1 e	(f 1 d 0)
2	d c 1 2	9 e 2 2	(f 2 2 3)
3	c a 8 f	4 b 3 7	(5 3 4 4)
4	0 8 9 6	1 3 2 d	(2 a 1 4)
5	7 6 0 3	0 8 d d	(9 a 7 7)
6	a 8 f d	9 a f 4	(6 a a a)
7	1 9 d 4	4 1 8 8	(a f 0 0)
8	3 8 7 1	8 7 7 b	(0 e 1 3)
9	e a d d	a 7 a 0	(4 b 3 a)
10	3 6 3 e	e 5 c 1	(c a e e)
11	4 6 5 8	3 7 d 0	(4 3 1 5)
12	3 0 7 1	b 5 c 9	(1 9 1 5)
13	c d a f	7 0 f e	(4 7 d f)
14	d f 1 3	1 8 6 d	(c e a 4)
15	f d 8 5	8 d 1 b	(4 6 1 5)
16	e 9 8 6	1 5 0 7 4	(8 c 6 e)
17	9 3 1 5 7	c 1 a e	(3 8 1 f)
18	8 7 d 4	7 7 3 5	(3 2 5 0)
19	d 0 8 6	8 6 9 4	(6 c d 9)
20	0 b b a	d 4 c 0	(4 1 2 9)
21	4 0 9 f	8 0 3 d	(f a e 1)
22	0 3 0 7	a 9 0 8	(3 4 c 5)
23	1 2 d 5	b e b c	(9 9 8 9)
24	f 7 1 c	b 5 0 5	(9 c b 5)
25	f c 6 1	b 4 c 6	(8 e 6 2)
26	f c a 4	4 6 b 0	(4 1 5 e)
27	2 f a 4	0 1 3 c	(7 9 d 6)
28	c c a 8	8 f 6 8	(2 c f 4)
29	e e c c	4 0 7 a	(9 f 7 6)
30	b f 5 6	c c b d	(a 5 b f)
31	f c 0 b	4 0 d 8	(d 0 d 6)
32	c 1 3 0	5 a 5 5	(b c d c)

№	X	K	Ответ
33	0 8 b f	6 0 c 2	(c d d f)
34	e 1 8 7	b e d 6	(3 2 1 4)
35	a 6 3 8	7 c 8 2	(9 9 5 2)
36	b 2 c 1	b 3 c 8	(6 b f 2)
37	d e 8 e	a d a 4	(4 3 2 b)
38	9 1 4 8	f e 8 9	(0 4 7 8)
39	9 d 0 a	8 a e 3	(5 b b d)
40	4 9 e 4	0 9 e e	(0 d f c)

4. Выполните преобразование столбца () с помощью операции MixColumns() для стандарта AES. Ответ запишите в виде (xx xx xx xx), где x – шестнадцатеричное число.

№	Исходный столбец	Номер столбца	Результат
1	1F 7B 59 AD	1	(47 AF 3A 42)
2	FA 77 76 82	2	(82 0C FC 0B)
3	AB CA 77 82	3	(FD 3F 12 44)
4	35 35 A2 7C	4	(EB DE DB 30)
5	6C 0D 5D AE	1	(3C 3F 32 A3)
6	D9 E2 B9 87	2	(AA 51 C0 3E)
7	40 52 76 16	3	(16 68 C4 C8)
8	0F 63 5F 2C	4	(C8 04 A6 75)
9	C9 7D 14 19	1	(03 16 B7 1B)
10	C9 7C 90 5E	2	(C3 C4 6C 10)
11	C0 67 82 64	3	(D4 F7 14 76)
12	35 61 01 EE	4	(26 1A 7F F8)
13	B0 E2 3D 38	1	(43 10 60 64)
14	49 0D D2 6D	2	(3A 53 4C DE)
15	39 64 13 69	3	(A4 AD C0 EE)
16	05 56 FA 51	4	(5B ED 4F 01)
17	C4 06 75 41	1	(AD 16 EB A6)
18	AB F0 42 7D	2	(79 EB 58 AE)
19	0F 3B B2 EB	3	(0A 5F 6D 55)
20	22 52 DB CD	4	(A4 3D 91 6E)

5. Регистр сдвига с линейной обратной связью задан полиномом P. Сформируйте последовательность длиною 2n битов, где n – размер РСЛОС, если изначально в регистре содержится число A.

№	Полином	Начальное заполнение (число A)	Ответ
1	x^4+x+1	12	1110101100
2	x^4+x+1	9	1101011001
3	x^4+x+1	14	0100011110
4	x^5+x^2+1	28	001000111100
5	x^5+x^2+1	25	111101011001
6	x^5+x^2+1	13	011110101101
7	x^5+x^2+1	19	111010110011
8	x^6+x+1	57	110001011111001
9	x^6+x+1	35	111100101000011
10	x^6+x+1	62	010000011111110
11	x^6+x+1	21	001111110101011
12	x^7+x+1	31	0000100000011111
13	x^7+x+1	40	1001111000101000
14	x^7+x+1	49	0111010010110001
15	x^7+x+1	61	1011000110111101
16	x^7+x+1	77	1001010111001101
17	x^7+x^3+1	123	1001101001111011
18	x^7+x^3+1	88	1111000111011000
19	x^7+x^3+1	56	1010011110111000
20	x^7+x^3+1	45	0101010000101101
21	x^7+x^3+1	126	0111000011111110
22	$x^8+x^4+x^3+x^2+1$	226	1101001000111000010
23	$x^8+x^4+x^3+x^2+1$	202	0011011101111001010
24	$x^8+x^4+x^3+x^2+1$	120	0011001110011111000
25	$x^8+x^4+x^3+x^2+1$	65	0101101101010000001
26	$x^8+x^4+x^3+x^2+1$	134	1101101111100001110
27	$x^8+x^4+x^3+x^2+1$	239	0110010111111101111
28	x^9+x^4+1	364	0100011101010110110 0
29	x^9+x^4+1	291	0010001000110010001 1
30	x^9+x^4+1	498	0100100110111111100 10
31	x^9+x^4+1	78	1010000101000100111 0

№	Полином	Начальное заполнение (число A)	Ответ
32	x^9+x^4+1	62	00110011101000111110
33	x^9+x^4+1	247	11111111000011110111
34	$x^{10}+x^3+1$	877	00110000000001101101101
35	$x^{10}+x^3+1$	377	00100101011001011111001
36	$x^{10}+x^3+1$	134	00111001011000100000110
37	$x^{10}+x^3+1$	474	01010110000101111011010
38	$x^{10}+x^3+1$	1022	01110000000011111111110
39	$x^{10}+x^3+1$	242	01101110110000111110010
40	$x^{11}+x^2+1$	5	01000000001000000000000101
41	$x^{11}+x^2+1$	157	00100101110100001001110l
42	$x^{11}+x^2+1$	805	11011111011000110010010l
43	$x^{11}+x^2+1$	1384	10000001100101010110l000
44	$x^{11}+x^2+1$	426	00001110000000011010l010
45	$x^{11}+x^2+1$	594	11110110001100100l010010

6. Для алгоритма шифрования A5/1 содержимое регистров равно: R1, R2, R3. Какие регистры будут работать на сдвиг в режиме stop-and-go?

№	R1	R2	R3	Ответ
1	$2604C_X$	$3002F8_X$	$54832A_X$	R1 R2 R3
2	$4C098_X$	$2005F0_X$	290654_X	R2 R3
3	$4C098_X$	$000BE1_X$	$520CA9_X$	R1 R2
4	18131_X	$0017C2_X$	$520CA9_X$	R1 R2 R3
5	30263_X	$002F84_X$	241953_X	R1 R3
6	$0CA72_X$	$14ED46_X$	$75DAE4_X$	R1 R3
7	$194E4_X$	$14ED46_X$	$6BB5C8_X$	R2 R3
8	$194E4_X$	$29DA8D_X$	$576B91_X$	R1 R2 R3
9	$329C9_X$	$13B51B_X$	$2ED723_X$	R1 R2 R3
10	65393_X	$276A37_X$	$5DAE47_X$	R1 R3
11	28223_X	$0E53F9_X$	$3B1D5E_X$	R1 R2
12	50447_X	$1CA7F2_X$	$3B1D5E_X$	R2 R3
13	504470_X	$394FE5_X$	$763ABC_X$	R1 R3
14	$2088E_X$	$394FE5_X$	$6C7578_X$	R2 R3

15	$2088E_X$	$329FCA_X$	$58EAF0_X$	R1 R3
№	R1	R2	R3	Ответ
16	30772_X	$39A09A_X$	$33CE06_X$	R1 R3
17	$60EE4_X$	$39A09A_X$	$679C0C_X$	R1 R2
18	$41DC8_X$	334134_X	$679C0C_X$	R1 R3
19	$03B91_X$	334134_X	$4F3818_X$	R2 R3
20	$03B91_X$	268268_X,	$1E7031_X$	R2 R3
21	$7D009_X$	$0EA95B_X$	$137BDF_X$	R1 R2 R3
22	$7A013_X$	$1D52B6_X$	$26F7BE_X$	R1 R2
23	74026_X	$3AA56D_X$	$26F7BE_X$	R1 R2 R3
24	74026_X	$354ADA_X$	$4DEF7C_X$	R1 R2
25	$6804D_X$	$2A95B4_X$	$4DEF7C_X$	R2 R3
26	54868_X	$2C8656_X$	$70277A_X$	R2 R3
27	54868_X	$190CAD_X$	$604EF5_X$	R2 R3
28	54868_X	$32195B_X$	$409DEB_X$	R1 R2
29	$290D0_X$	$2432B6_X$	$409DEB_X$	R1 R2
30	$521A1_X$	$08656D_X$	$409DEB_X$	R1 R3
31	$51F42_X$	$13ED86_X$	$02EA14_X$	R1 R2
32	$23E84_X$	$27BD0D_X$	$02EA14_X$	R1 R3
33	$47D08_X$	$0FB61B_X$	$05D428_X$	R1 R2 R3
34	$0FA10_X$	$1F6C36_X$	$0BA850_X$	R1 R2 R3
35	$1F421_X$	$1F6C36_X$	$1750A0_X$	R1 R3

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шеннон К. Теория связи в секретных системах // http://www.enlight.ru/crypto/articles/shannon/shann_i.htm

2. Шнайер Б. Прикладная криптография: протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2002. – С. 648.

3. Асосков А.В. и др. Поточные шифры // А.В. Асосков, М.А. Иванов, А.А. Мирский, А.В. Рузин, А.В. Сланин, А.Н. Тютвин. – М.: КУДИЦ-ОБРАЗ, 2003. – 336 с.

4. Основы функции хэширования [Электронный ресурс], 27.02.2015. – http://www.cryptofaq.ru/my_hash_post.html, свободный.

5. Взлом хеш-функций (2004 – 2006): как это было и что теперь делать? [Электронный ресурс], 27.02.2015. – http://habrahabr.ru/blogs/infosecurity/50434, свободный.

6. Столлингс В. Криптография и защита сетей: принципы и практика: пер. с англ. – М.: Издательский дом «Вильямс», 2001.

7. Панасенко С., Алгоритмы шифрования: специальный справочник. – СПб.: БХВ-Петербург, 2009. – 576 с.

8. Чмора А.Л. Современная прикладная криптография. – 2-е изд. – М.: Гелиос АРВ, 2002.

9. Бабенко Л.К. Ищукова Е.А. Современные алгоритмы блочного шифрования и методы их анализа. – М.: Гелиос АРВ, 2006.

10. Saarien M.-J. A Chosen Key Attack Against the Secret S-boxes of GOST // http://www.m.-js.com – Helsinki University of Technology, Finland.

11. Зензин О.С., Иванов М.А. Стандарт криптографической защиты – AES. Конечные поля. – М.: КУДИЦ-ОБРАЗ, 2002.

12. ТК-26 Блочные шифры [Электронный ресурс]. – Режим доступа: http://www.tc26.ru/standard/draft/bsh.php, 28.01.2015

13. Бабенко Л.К., Ищукова Е.А. Криптографические методы и средства обеспечения информационной безопасности: учебное пособие. – Таганрог: Изд-во ТТИ ЮФУ, 2011. – 148 с.

14. Жуков А.Е. Легковесная криптография. Ч. 1 // Вопросы кибербезопасности. – 2015. – №1(9). – С. 26 – 43.

15. Toru Akishita and Harunaga Hiwatari Very Compact Hardware Implementations of the Blockcipher CLEFIA [Электронный ресурс]. – Режим доступа: http://www.sony.co.jp/Products/cryptography/clefia/ download/data/clefia-hw-compact-20110615.pdf, свободный.

16. CLEFIA the 128-bit blockcipher [Электронный ресурс]. – Режим доступа: www.sony.net/clefia/, свободный.

17. Грушо А.А., Тимонина Е.Е., Применко Э.А. Анализ и синтез криптоалгоритмов: курс лекций. – Йошкар-Ола: Изд-во МФ МОСУ, 2000.

18. Matsui M. Linear Cryptanalysis Method for DES Cipher, Advances in Cryptology – EUROCRYPT'93, Springer-Verlag, 1998. – P. 386.

19. Biham E., Shamir A. Differential Cryptanalysis of the Full 16-round DES, Crypto'92, Springer-Velgar, 1998. – P. 487.

20. Biham E., Shamir A. Differential Cryptanalysis of DES-like Cryptosystems, Extended Abstract, Crypto'90, Springer-Velgar, 1998. – P. 2.

21. Courtois N., Pieprzyk J. Cryptanalysis of block ciphers with overdefined systems of equations // ASIACRYPT. – 2002. – P. 267–287.

22. Courtois N., Klimov A., Patarin J., Shamir A. Efficient algorithms for solving overdefined systems of multivariate polynomial equations // EUROCRYPT, 2000. – P. 392–407.

23. Courtois N., Gregory V. Bard. Algebraic Cryptanalysis of the Data Encryption Standard// 11-th IMA Conference. – 2007. – P. 152–169.

24. Kleiman E. The XL and XSL attacks on Baby Rijndael //http://orion.math.iastate.edu/dept/thesisarchive/MS/EKleimanMSSS05.pdf.

25. Маро Е.А. Алгебраический криптоанализ упрощенного алгоритма шифрования Rijndael // Известия ЮФУ. Техниче-

ские науки. Тематический выпуск «Информационная безопасность». – 2009. – №11. – С. 187–199.

26. http://csrc.nist.gov/encryption/aes – Daemen J., Rijmen, V. AES Proposial: Rijndael.

27. http://www.csberkeley.edu~daw/papers/advslide-ec00.ps - Birukov A., Wagner D. Advanced Slide Attacks.

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

ДЛЯ ЗАМЕТОК

Учебное издание

Бабенко Людмила Клементьевна
Ищукова Евгения Александровна

Криптографическая защита информации: симметричное шифрование

по курсу

Криптографические методы защиты информации

Учебное пособие

Ответственный за выпуск Ищукова Е.А.
Редактор Надточий З.И.
Корректор Селезнева Н.И.

Подписано в печать 15.10.2015
Заказ № 184 Тираж 50 экз.
Формат 60x84 1/16. Усл. печ. л. – 14,0. Уч.-изд. л. – 13,38.

Издательство Южного федерального университета
344091, г. Ростов-на-Дону, пр. Стачки, 200/1.
Тел. (863)2478051.
Отпечатано в Секторе обеспечения полиграфической продукцией кампуса в г. Таганроге отдела полиграфической, корпоративной и сувенирной продукции ИПК КИБИ МЕДИА ЦЕНТРА ЮФУ.
ГСП 17 А, Таганрог, 28, Энгельса, 1.
Тел. (8634)371717.