Московский Государственный Технический Университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ-8

Жуков А.Е.

Системы блочного шифрования

Пособие по курсу

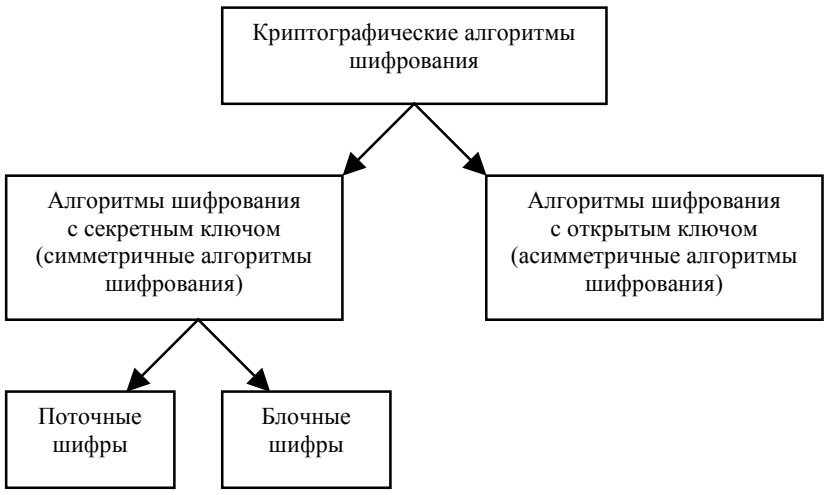
«Криптографические методы защиты информации»

(IX семестр)

Системы блочного шифрования

В основе подавляющего большинства систем защиты информации лежит использование того или иного криптографического преобразования информации. Неформально говоря, это пара взаимно-обратных преобразований $E(X)=Y$, $D(Y)=X$, где X – открытая информация, а Y – соответствующая ей шифрованная информация. Преобразования E и D называются *криптографическими алгоритмами*: E – алгоритм *шифрования* (*encryption*), D – алгоритм *расшифрования* (*decryption*). При этом для законного пользователя реализация обоих алгоритмов E и D не представляет трудностей, в то время как для злоумышленника, имеющего доступ к информации Y, получение информации X является или абсолютно невозможным или практически невозможным, т.е. требует таких временных и стоимостных затрат, что становится практически нереальным.

В настоящее время общепризнанным является подразделение криптографических алгоритмов шифрования (называемых также просто шифрами) на следующие основные категории:



Последнее подразделение неформально можно понимать следующим образом: поточные шифры подвергают криптографическому преобразованию отдельные знаки шифруемой информации, в то время как блочные шифры преобразуют сразу большие группы знаков. Так как основной областью приложения теории булевых функций и преобразований в настоящее время являются симметричные алгоритмы шифрования, далее в этой книге будем рассматривать только этот класс криптографических алгоритмов.

Далее в этой главе будут рассмотрены основные понятия, используемые для описания работы блочных шифров, примеры типовых узлов, входящих в их конструкцию, а также наиболее распространенные схемы построения блочных шифров. В заключение приводится детальное описание трех алгоритмов блочного шифрования: DES, AES и ГОСТ 28147-89. Первые два являются федеральными стандартами США (приняты в 1976 и 2001 гг. соответственно). ГОСТ 28147-89 – Государственный стандарт СССР и России. Описание большого числа блочных шифров можно найти в энциклопедической монографии [7] , а также в работах [2] и [18].

§ 1. Основные определения и понятия

Системами блочного шифрования будем называть устройства или алгоритмы, которые преобразуют информацию, заданную в виде элемента p конечного множества P в элемент c конечного множества C, используя при этом ключ k, являющийся элементом конечного множества K. Эти системы относятся к классу конечных детерминированных систем шифрования. ***Детерминированная система шифрования*** представляет собой четверку $\Pi = (P,C,K,F)$, где P, C и K – множества, называемые, соответственно, пространством входов или ***открытых текстов*** (*plaintext*), пространством выходов или ***шифртекстов*** *(ciphertext*) и пространством ***ключей*** (*key*), а F является отображением F: $P \times K \to C$, при этом для каждого фиксированного $k \in K$ отображение $F_{(k)}: P \to C$, задаваемое формулой $F_{(k)}(p) = F(p,k)$, является инъективным отображением (условие, необходимое для однозначного расшифрования шифртекста). Из сложившейся практики система блочного шифрования (или просто ***блочный шифр***) должна обладать и другими свойствами. Так пространства P и C обычно представляют собой множества двоичных слов фиксированной длины n; в этом случае элементы этих пространств называются также ***информационными блоками*** длины n. Таким образом, $P = C = \mathbb{Z}_2^n$, а в качестве ключевого пространства, как правило, выбирается множество $\mathbb{Z}_2^k$. В этом случае и ключи и информационные блоки удобно рассматривать как двоичные векторы из соответствующих векторных

пространств, а шифрующее отображение F задавать с помощью различных операций над бинарными векторами. Указанные особенности приводят к тому, что для исследования блочных шифров помимо традиционных методов криптографического анализа применяются специально разработанные методы, связанные со спецификой их строения, в том числе и со свойствами используемых в этих алгоритмах булевых функций и преобразований.

Пусть имеется семейство систем блочного шифрования $\Pi_i = (P_i, C_i, K_i, F_i)$, $i = 1,\ldots,R,$ и при этом все системы Π_i имеют одно и то же пространство P в качестве пространств входов и выходов, т.е. $P_i = C_i = P$ для всех i, а каждое отображение $F_i : P \times K_i \to P$ определено своим ключевым элементом $k_i \in K_i$. Тогда на базе этого семейства с помощью операции композиции отображений можно построить блочный шифр, задаваемый отображением $F : P \times (K_1 \times \ldots \times K_R) \to P$, где $F = F_R \circ F_{R-1} \circ \ldots \circ F_2 \circ F_1$. Такой шифр называется ***композиционным*** или ***каскадным*** шифром для семейства шифров F_i, ключом его является вектор $(k_1,\ldots k_R) \in K_1 \times \ldots \times K_R$. В иностранной литературе такой шифр часто называется ***послойным*** (*layered*).

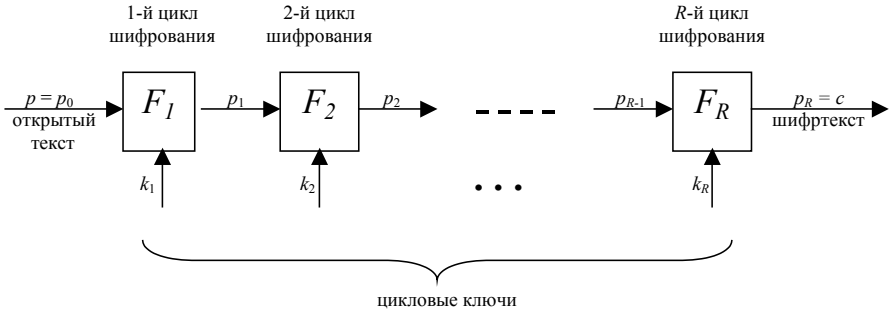


Рис. 1

На рис.1 приводится схема работы композиционного шифра. Отображение F_i называется ***i-м циклом шифрования*** (*round*) или ***i-м цикловым шифр-преобразованием*** композиционного шифра F. Информационный блок $p_i = F_i(p_{i-1}, k_i)$ называется i–м ***промежуточным информационным блоком***. В качестве информационного блока p_0 выступает открытый текст p. Шифртекстом является значение информационного блока p_R. Ключ k_i называется ***ключом i-го цикла шифрования*** или просто ***цикловым ключом***, если i определено. Во многих системах цикловые ключи получаются из ключа всей системы (величина которого, как правило, существенно меньше суммарного

объема всех цикловых ключей) с помощью специального ***алгоритма выработки цикловых ключей*** (*key scheduling algorithm*). Если для композиционного шифра $F = F_R \circ F_{R-1} \circ \ldots \circ F_2 \circ F_1$ ключевые пространства K_i совпадают, т.е. $K_1 = \ldots = K_R = K$ и при этом существует отображение $F: P \times K \to P$, такое, что для всех $i=1,\ldots, R$ отображение $F_i = F$, то такой композиционный блочный шифр называется ***итерационным шифром***.

Отображение пространства P в себя, осуществляемое итерационным шифром, представляет собой композицию одного и того же криптографического отображения, но с разными ключами. Композиционные, а особенно итерационные шифры являются наиболее распространенным типом блочных шифров среди реально существующих или предложенных теоретически, что объясняется удобством как их программной реализации, так и их реализации с помощью интегральных схем.

§ 2. Основные узлы блочных шифров. S-блоки и P-блоки

В теоретическом плане, лежащая в основе большинства итерационных блочных шифров идея состоит в построении криптографически стойкой системы путем последовательного применения относительно простых криптографических преобразований. Принцип многоразового шифрования с помощью простых криптографических преобразований был впервые предложен Шенноном в работе [6], где он использовал с этой целью преобразования перестановки и подстановки. Первое из этих преобразований переставляет отдельные символы преобразуемого информационного блока, а второе – заменяет каждый символ (или группу символов) из преобразуемого информационного блока другим символом из того же алфавита (соответственно группой символов того же размера и из того же алфавита). Узлы, реализующие эти преобразования, называются, соответственно, ***P-блоками*** (*P-box, permutation box*) и ***S-блоками*** (*S-box, substitution box*). В подавляющем большинстве случаев входные и выходные информационные блоки для этих узлов являются двоичными векторами, т.е. отдельные символы являются битами. Если число входных и выходных символов P-блока или S-блока совпадает и равно n, то они называются, соответственно, P-блоком размера $n \times n$ и S-блоком размера $n \times n$ или еще короче – $P_{n \times n}$-блоком и $S_{n \times n}$-блоком. Частным случаем $P_{n \times n}$-блока является легко реализуемая на компьютере операция циклического сдвига двоичного набора длины n.

Схемы, построенные из узлов, осуществляющих эти преобразования, называются ***SP-сетями*** (*SP*-nets). Использование таких преобразований позволяет получать отображения, удовлетворяющие двум свойствам, которыми по

Шеннону должны обладать криптографически стойкие отображения: свойству рассеивания и свойству перемешивания. Под ***рассеиванием*** (*diffusion*) понимается распространение влияния одного знака открытого текста или ключа на много знаков шифртекста. Это свойство позволяет скрыть статистическую зависимость между знаками открытого текста, а также не позволяет восстанавливать неизвестный ключ по частям. В свою очередь ***перемешивание*** (*confusion*) означает использование таких преобразований, которые усложняют обнаружение связи между открытым и шифрованным текстами с помощью статистических и иных методов.

Пример. Рассмотрим $P_{8\times8}$-блок, переставляющий координаты входного 8-битного вектора в соответствии с перестановкой $\begin{pmatrix}1&2&3&4&5&6&7&8\\3&8&6&1&7&2&4&5\end{pmatrix}$ (см. рис.2). Так, например, входной вектор (11100010) преобразуется в вектор (00110101). В описаниях алгоритмов P-блоки задаются обычно перечислением номеров входных битов, соответствующих первому, второму и т.д. выходным битам. Так наш P-блок может быть задан строкой 46178352.

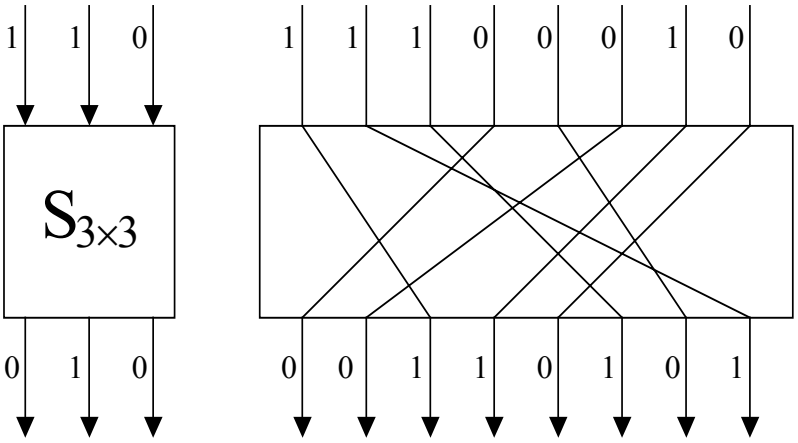


Рис. 2

На том же рисунке изображен $S_{3\times3}$-блок, осуществляющий замену входного 3-битного вектора на другой 3-битный вектор в соответствии с подстановкой $\begin{pmatrix}000&001&010&011&100&101&110&111\\101&000&110&100&111&011&010&001\end{pmatrix}$. Обычно в таблицах подстановок двоичные векторы представляются своими числовыми значениями. В этом слу-

чае наш S-блок может быть представлен в виде подстановки $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 0 & 6 & 4 & 7 & 3 & 2 & 1 \end{pmatrix}$ или еще короче – только нижней строкой: 50647321.

На практике используются также такие P-блоки и S-блоки, у которых число входных и выходных битов не совпадает. Если узел преобразует n-битный вектор в m-битный то говорится, что он имеет размер $n \times m$. P-блок для которого $n>m$ называется ***усеченной перестановкой***, в случае $n<m$ он называется ***расширяющей перестановкой***. Соответственно в случае $n>m$ S-блок называется ***сжимающим***, а в случае $n<m$ – ***растягивающим***.

Пример. Рассмотрим $P_{8\times12}$-блок, являющийся расширяющей перестановкой, которая определяется строкой 123434567878. Схематически этот P-блок изображен на рис.3.

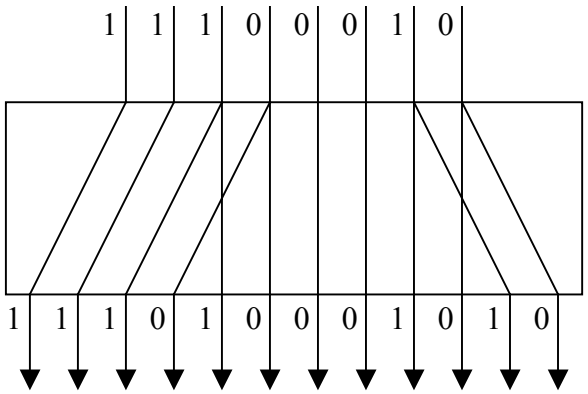


Рис. 3

В современных системах блочного шифрования цикловые шифрпреобразования строятся в основном с использованием арифметических операций, при которых двоичные наборы рассматриваются как числа или как элементы какой-либо другой алгебраической структуры, и линейных операций над двоичными векторами, в том числе операций перестановки компонент в двоичном векторе (P-блоки) и, в частности, операций циклического сдвига двоичного вектора. Операция сложения векторов часто используется для смешения ключевых элементов с перерабатываемым информационным блоком. Очень важным элементом являются блоки подстановки (S-блоки), осуществляющие нелинейные преобразования для двоичных векторов небольшой размерности. Как правило – это единственные нелинейные преобразования, уча-

ствующие в образовании циклового шифра, и стойкость всей системы во многом зависит от их свойств.

Отметим, что системы шифрования, построенные как композиция большого числа слабых криптографических преобразований, становятся неустойчивыми, если становятся известными значения каких-либо промежуточных информационных блоков. Как правило, соответствующие технические устройства не позволяют располагать такой информацией, поэтому при анализе стойкости композиционных шифров использование промежуточной информации считается невозможным [9].

§ 3. Схемы Фейстеля

Это самая популярная на сегодняшний день схема построения итерационного шифра. Большинство опубликованных в настоящее время блочных шифров так или иначе используют в своей структуре схему Фейстеля.

Итерационный шифр $F = F_R \circ F_{R-1} \circ \ldots \circ F_2 \circ F_1$ называется ***шифром Фейстеля*** (Feistel) или ***схемой Фейстеля***, если

1. $P = T \times T$, т.е. любой информационный блок рассматривается как упорядоченная пара «полублоков» меньшего размера: $p = (l,r)$, где r, l лежат в T;
2. Цикловое шифрпреобразование F_i задается формулой
$$F_i(l, r) = (r, l \otimes f_i(r)), \qquad (1)$$
где $f_i: T \times K_i \to T$ – функция, зависящая от циклового ключа k_i, а $\otimes$ – означает некоторую обратимую операцию, относительно которой множество T замкнуто.

Преобразование $F(l, r) = (r, l \otimes f(r))$ называется ***преобразованием Фейстеля*** (см. рис.4). Как правило, в качестве множества T выбирается векторное пространство $\mathbb{Z}_2^m$, а в качестве $\otimes$ – операция векторного сложения в пространстве $\mathbb{Z}_2^m$, обозначаемая обычно как + или $\oplus$. В этом случае преобразование Фейстеля может быть реализовано с помощью двух m-разрядных двоичных регистров, соединенных в схему, изображенную на рис.5.

Выбор преобразования (1) в качестве одноциклового шифрпреобразования объясняется тем, что преобразование указанного вида является подстановкой на $\mathbb{Z}_2^{2m}$, а композиционный шифр, образованный из таких преобразований, при случайном выборе цикловых ключей k_i представляет собой псевдослучайную подстановку на $\mathbb{Z}_2^{2m}$, имеющую характеристики

псевдослучайную подстановку на $\mathbb{Z}_2^{2m}$, имеющую характеристики весьма близкие к случайной подстановке (см. [19], [23], [27]).

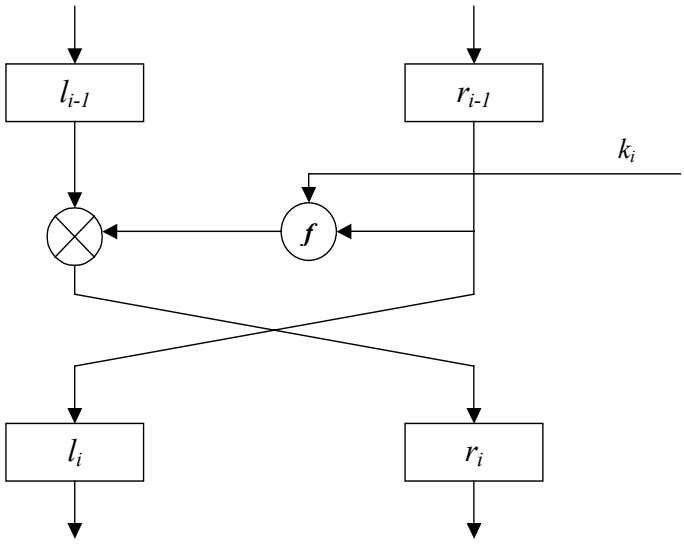


Рис. 4

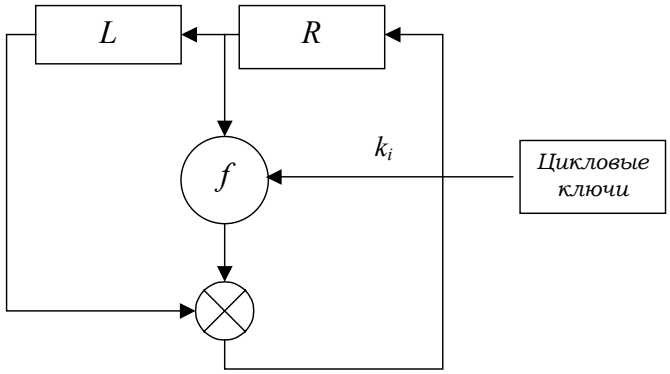


Рис. 5

Обобщенный шифр Фейстеля был определен в работах [14],[25] как итерационный блочный шифр F, цикловое шифрпреобразование которого осуществляет отображение $F_f^{(1)}: \mathbb{Z}_2^{rm} \to \mathbb{Z}_2^{rm}$ по формуле

$$F_f^{(1)}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_r) = (\mathbf{b}_2 \oplus f_{k_i}(\mathbf{b}_1), \mathbf{b}_3, \ldots, \mathbf{b}_r, \mathbf{b}_1), \qquad (2)$$

где $\mathbf{b}_i \in \mathbb{Z}_2^m$; $f_{k_i}: \mathbb{Z}_2^m \to \mathbb{Z}_2^m$; $\oplus$ – векторное сложение в пространстве $\mathbb{Z}_2^m$.

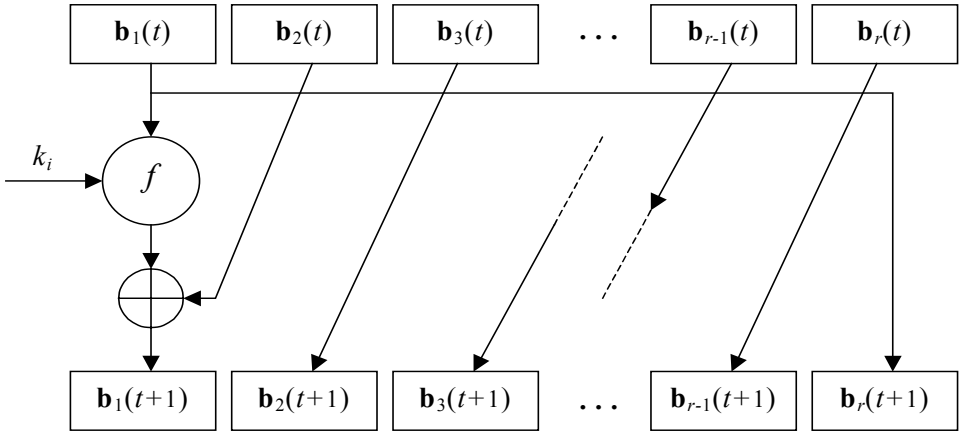


Рис. 6

Схематично это преобразование изображено на рис.6. На рис.7 изображена реализация этого преобразования с помощью двоичных m-разрядных регистров.

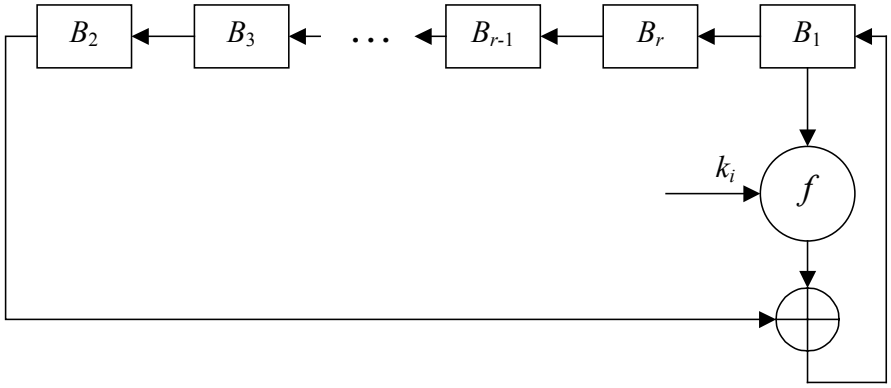


Рис. 7

В работе [27] шифр, задаваемый преобразованием (2), назван ***обобщенным шифром Фейстеля 1-го типа***, там же были введены обобщенные шифры Фейстеля 2-го и 3-го типов. В ***обобщенном шифре Фейстеля 2-го типа*** цикловой шифр осуществляет отображение $F_f^{(2)}: \mathbb{Z}_2^{rm} \to \mathbb{Z}_2^{rm}$, r – четное, по формуле:

$$F_f^{(2)}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_r) = \\ = (\mathbf{b}_2 \oplus f_1(\mathbf{b}_1), \mathbf{b}_3, \mathbf{b}_4 \oplus f_3(\mathbf{b}_3), \ldots, \mathbf{b}_r \oplus f_{r-1}(\mathbf{b}_{r-1}), \mathbf{b}_1), \tag{3}$$

где $\mathbf{b}_i \in \mathbb{Z}_2^m$; f_i – функции, отображающие $\mathbb{Z}_2^m \times K \to \mathbb{Z}_2^m$; $\oplus$ – векторное сложение в пространстве $\mathbb{Z}_2^m$. Схематично это преобразование изображено на рис.8.

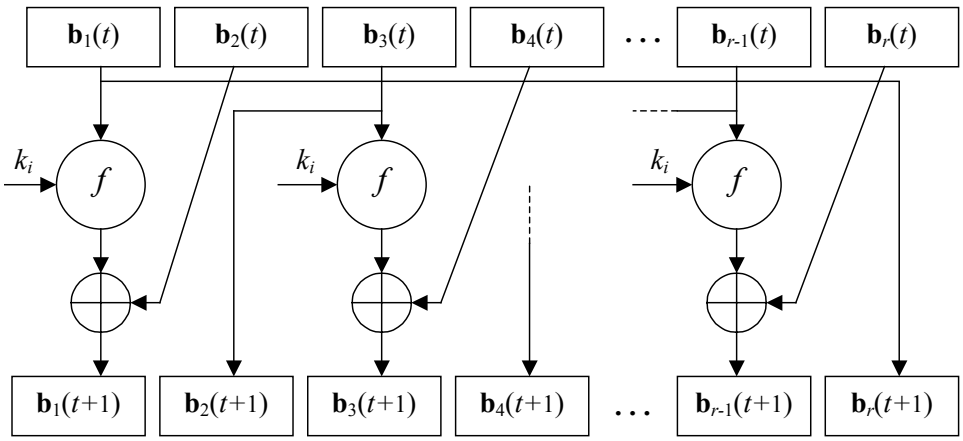


Рис. 8

На рис.9 изображена реализация этого преобразования с помощью двоичных m-разрядных регистров.

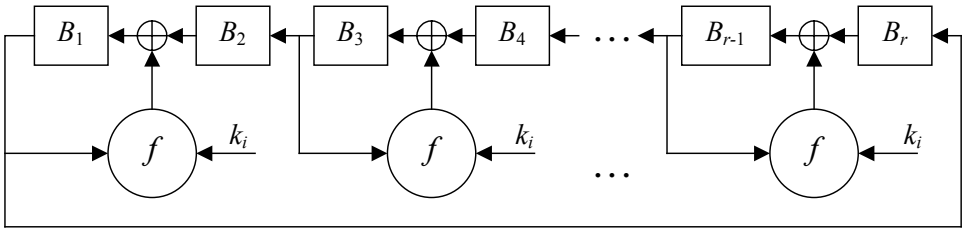


Рис. 9

Аналогично, ***обобщенный шифр Фейстеля 3-го типа*** определяется как композиционный шифр, цикловой шифр которого осуществляет отображение $F_f^{(3)}: \mathbb{Z}_2^{rm} \to \mathbb{Z}_2^{rm}$, вида

$$F_f^{(3)}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_r) = (\mathbf{b}_2 \oplus f_1(\mathbf{b}_1), \mathbf{b}_3 \oplus f_2(\mathbf{b}_2), \ldots, \mathbf{b}_r \oplus f_{r-1}(\mathbf{b}_{r-1}), \mathbf{b}_1), \qquad (4)$$

где $\mathbf{b}_i$, f_i и $\oplus$ – такие же, как и определенные выше. Схематично это преобразование изображено на рис.10. На рис.11 изображена реализация этого преобразования с помощью двоичных m-разрядных регистров. Разумеется, все функции f_i в обобщенных шифрах Фейстеля любого типа зависят от соответствующего циклового ключа.

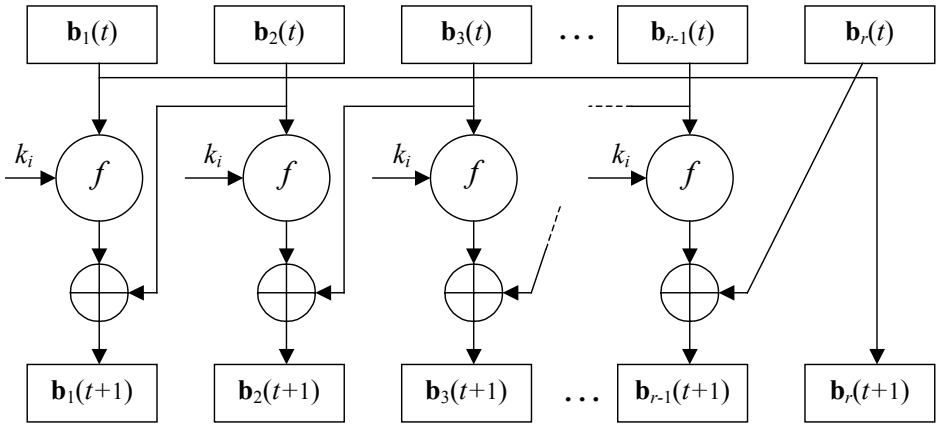


Рис. 10

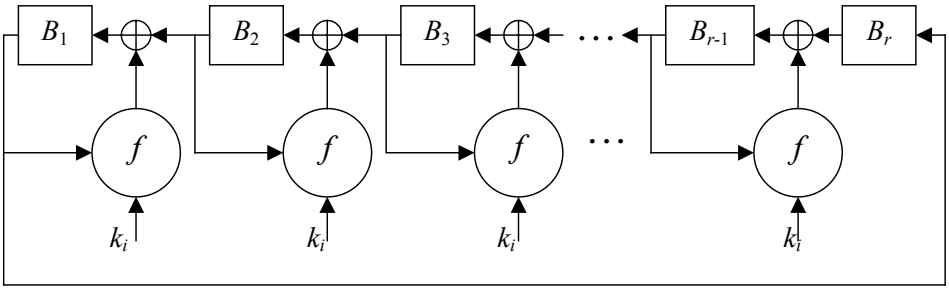


Рис. 11

Возвращаясь к классическому преобразованию Фейстеля (рис.4), отметим, что осуществляемая под конец этого преобразования перестановка левого и правого полублоков не играет никакой роли с точки зрения криптографической стойкости алгоритма. По-видимому, эта перестановка является следствием изначальной реализации этого преобразования на аппаратном уровне с помощью сдвиговых регистров (рис.5). В связи с изменением элементной базы, произошедшим за последние десятилетия, указанная реализация в большой степени потеряла смысл. В связи с этим все чаще используются программно реализованные схемы Фейстеля, не использующие перестановку полублоков (см. рис.12).

Отметим, кроме того, растущую популярность ***неравновесных схем Фейстеля*** – схем в которых левые и правые полублоки информационного блока имеют разную длину. Естественно, для реализации их совместных преобразований приходится пользоваться сжимающими и расширяющими преобразованиями двоичных векторов.

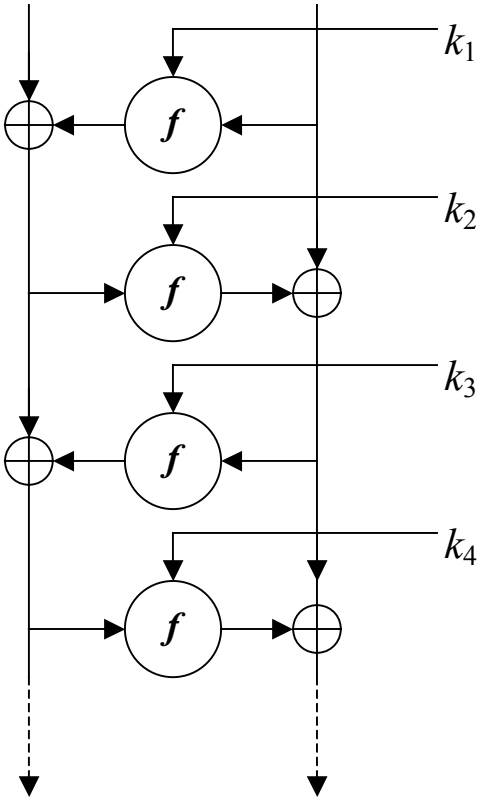


Рис. 12

§ 4. DES: Data Encryption Standard – стандарт США, 1977 г.

В 1973-74 гг. Национальное Бюро Стандартов США (NBS) опубликовало документы, содержащие требования к криптографическому алгоритму, который мог бы быть принят в качестве стандарта шифрования данных в государственных и частных учреждениях. В 1976 г. в качестве такового стандарта был утвержден алгоритм, разработанный фирмой IBM. В 1977 г. этот стандарт был официально опубликован и вступил в силу как федеральный стандарт шифрования данных – Data Encryption Standard или сокращенно ***DES*** [10].

Строго говоря, стандарт шифрования данных, понимаемый в широком смысле, помимо алгоритма шифрования должен содержать многочисленную дополнительную информацию: формат представления данных, режимы пользования, требования к технической реализации, правила, регулирующие работу с ключами и т.д. В свете этого сам шифрующий алгоритм довольно часто называют Data Encryption Algorithm – DEA. Под этим названием он фигурирует в материалах Американского Национального Института Стандартизации (ANSI). В публикациях Международной Организации Стандартизации (ISO) этот алгоритм получил название DEA-1. Кроме того, действующий в настоящее время стандарт шифрования помимо первоначального алгоритма DES содержит алгоритм TDEA (прежнее название – Triple DES или 3-DES). Тем не менее, DES остался наиболее употребимым названием описываемого алгоритма. Этому названию отдадим предпочтение и мы.

На протяжении последних 25 лет DES подвергался интенсивному и всестороннему анализу со стороны многочисленных исследователей. Несмотря на обнаруженные за это время слабости, а также проведенные в 1998 г. эксперименты по непосредственному нахождению ключа методом тотального опробования [1)], DES продолжает оставаться федеральным стандартом. Информацию о статусе DES как федерального стандарта, так и международного стандарта можно найти в работах [7], [10], [28] и в обзоре [5]. В работах [5], [7], кроме того содержится история создания DES, перспективы дальнейшего

[1)] В январе-феврале 1998 г. алгоритм DES был вскрыт с помощью распределенных вычислений в сети Internet. Около 22000 компьютеров разного класса (50000 CPU) были объединены в единую вычислительную сеть с помощью сети Internet. За 40 дней работы был найден ключ, на котором был зашифрован заданный текст. За это время было опробовано приблизительно 85% от общего числа ключей. Летом того же года специализированный компьютер EFF DES Cracker стоимостью 250000$ нашел ключ за 56 часов работы.

использования DES, а также весьма краткий обзор работ, посвященных анализу этого алгоритма.

В самом схематичном виде алгоритм DES представляет собой 16-цикловой итерационный блочный шифр Фейстеля с $\mathbb{Z}_2^{64}$ в качестве пространства входов-выходов и $\mathbb{Z}_2^{56}$ в качестве пространства ключей. Функция $\boldsymbol{f}$, участвующая в определении циклового шифра F_i , получает на вход правую часть информационного блока, представляющую собой 32-мерный двоичный вектор, и 48-битовый цикловой ключ. Получаемый в результате 32-мерный двоичный вектор складывается как элемент векторного пространства $\mathbb{Z}_2^{32}$ с левой частью информационного блока, после чего обе половины 64-битового информационного блока меняются местами. Прочие преобразования, а также алгоритм выработки 48-битовых цикловых ключей из 56-битового ключа системы служат для обеспечения необходимого перемешивания и рассеивания перерабатываемой информации, однако при анализе DES чаще всего играют не самую существенную роль.

Приведем теперь более детальное описание алгоритма DES. Общая схема вычислений, производимых алгоритмом DES, приведена на рис.13.

Все биты в информационных блоках нумеруются слева направо, начиная с 1: $a_1, a_2, a_3, \ldots$.

Алгоритм шифрования

Начальная перестановка **IP** (P-блок размера 64×64) осуществляет перестановку бит входного 64-битного информационного слова в соответствии со следующей таблицей [1)]:

Начальная перестановка IP:

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

[1)] Здесь и далее таблицы перестановок (P-блоков) следует читать как текст – слева направо, сверху вниз.

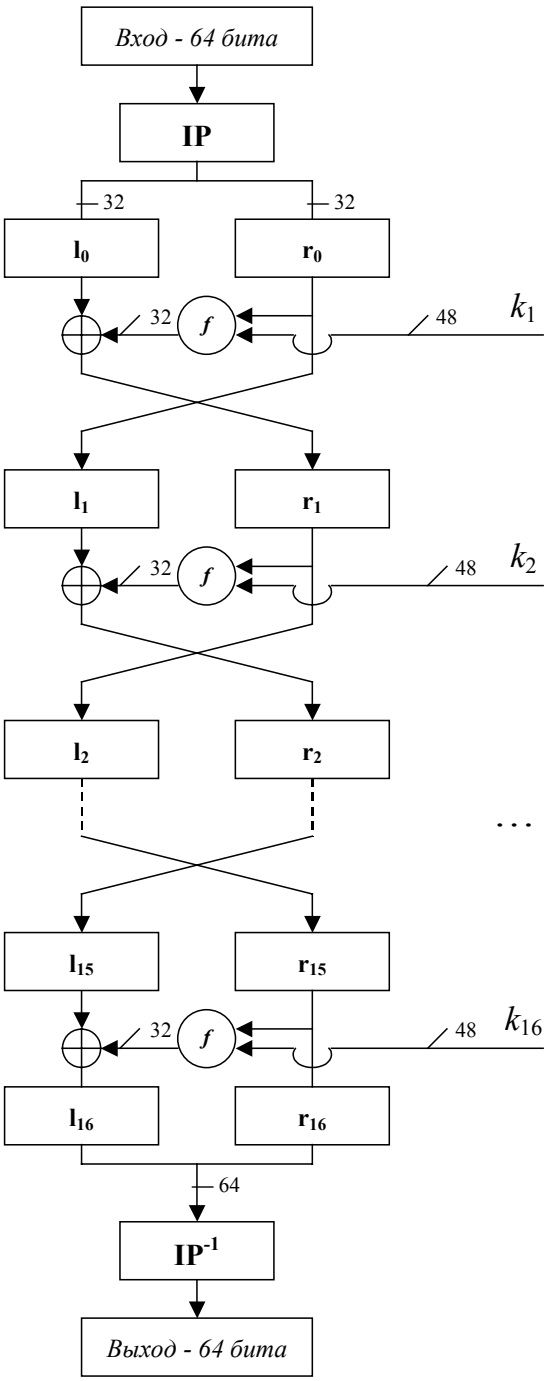

Вход - 64 бита
IP
32
32
l0
r0
32
f
48
k1
l1
r1
32
f
48
k2
l2
r2
. . .
l15
r15
32
f
48
k16
l16
r16
64
IP-1
Выход - 64 бита

Рис. 13

Как указывалось выше, перестановку бит информационного блока принято задавать перечислением номеров бит входа, соответствующих первому, второму и т.д. битам выхода. Так, в соответствии с таблицей, 1-м битом преобразованного блока будет 58-й бит входа, 2-м – 50-й бит входа и т.д., …, 64-м битом выхода будет 7-й бит входа.

Финальная перестановка $\mathbf{IP}^{-1}$ является обратной к перестановке **IP**. Она задается таблицей

Финальная перестановка IP $^{-1}$:

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

Заметим, что на вход финальной перестановки приходит информационный блок, полученный после последнего, 16-го цикла шифрования. В отличии от предыдущих 15 циклов, в конце этого цикла левый и правый полублоки информационного блока местами не меняются (см. рис….). Это необходимо для того, чтобы алгоритм расшифрования структурно совпадал с алгоритмом шифрования.

Функция $\boldsymbol{f}$ осуществляет отображение $\mathbb{Z}_2^{32} \times \mathbb{Z}_2^{48} \to \mathbb{Z}_2^{32}$ (см. рис.14). Это отображение образовано следующими преобразованиями:

1. 32 бита данных, составляющих полублок $\mathbf{r}_{i-1}$, поступают на вход расширяющей перестановки **E** (P-блок размера 32×48), который задается таблицей

   Расширяющая перестановка E

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

   Таким образом, выходом перестановки **E** является 48-битный вектор, полученный дублированием некоторых бит входа.

2. 48-битный вектор, полученный в п.1, побитно складывается с 48-битным цикловым ключом k_i .

3. 48-битный вектор, полученный в п.2, делится на восемь 6-битных слов. Каждое из них поступает на вход соответствующего $S_{6\times4}$-блока, которые обозначены как $\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_3}$, $\mathbf{S_4}$, $\mathbf{S_5}$, $\mathbf{S_6}$, $\mathbf{S_7}$, $\mathbf{S_8}$. Работа S-блоков будет описана ниже. Восемь 4-битных слов, являющихся выходами S-блоков, конкатенируются в 32-битный вектор.

4. 32-битный вектор, полученный в п.3, поступает на вход перестановки **P**, (P-блок размера 32×32), которая задается следующей таблицей:

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

Полученный на выходе перестановки **P** информационный блок является значением $\boldsymbol{f}(\mathbf{r}_{i-1}, k_i)$.

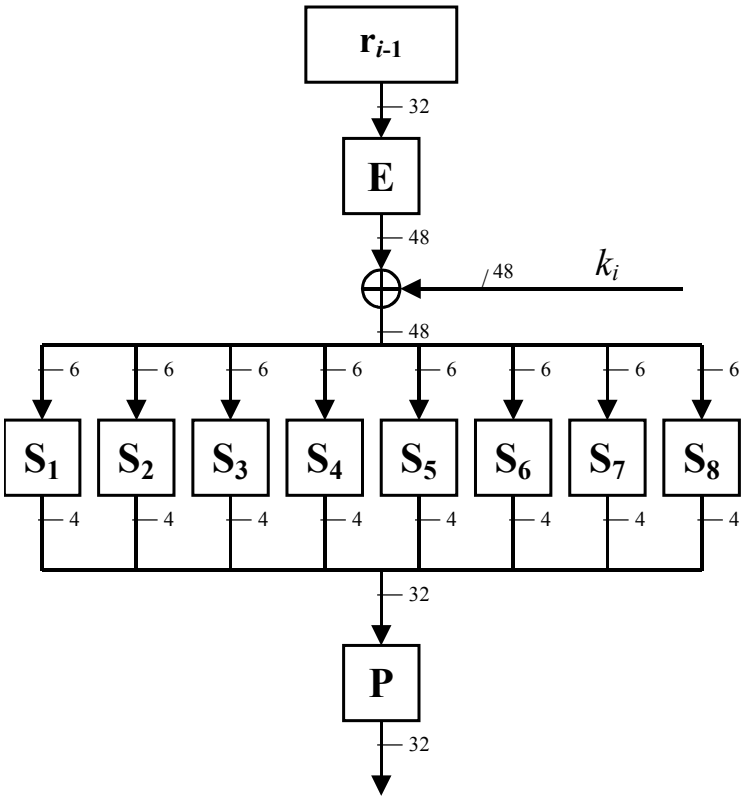


Рис. 14

Работа S-блоков $\mathbf{S_1} – \mathbf{S_8}$ может быть изображена следующей схемой:

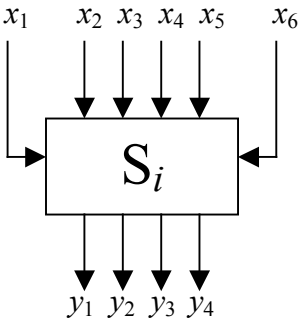


Рис.15

Первый и последний биты входа называются управляющими битами, средние четыре бита – преобразуемыми. В зависимости от значения пары управляющих битов $\mathbf{r} = (x_1, x_6)$, 4-битный набор $\mathbf{x} = (x_2, x_3, x_4, x_5)$ преобразуется в выходной 4-битный набор $\mathbf{y} = (y_1, y_2, y_3, y_4)$ в соответствии с таблицей по следующим правилам:

- номер управляющего набора $\mathbf{r} = (x_1, x_6)$ определяет номер строки в соответствующей таблице преобразований (верхняя строка имеет номер 0, нижняя – номер 3);
- вектор $\mathbf{x} = (x_2, x_3, x_4, x_5)$ (обозначаемый номером соответствующего набора) определяет номер столбца; самый левый столбец имеет номер 0, самый правый – 15;
- выходом является 4-битный $\mathbf{y} = (y_1, y_2, y_3, y_4)$, соответствующий двоичной записи числа, стоящего в таблице на соответствующем месте.

Таблицы S-блоков $\mathbf{S_1} – \mathbf{S_8}$ приведены на следующей странице.

Пример. Рассмотрим, как пятый S-блок преобразует вход (110110). В этом случае управляющий набор $\mathbf{r} = (1,0)$, а преобразуемый набор $\mathbf{x} = (1,0,1,1)$. В таблице «S-блок номер 5» в строке с номером 2 (третья строка сверху) выбираем столбец с номером 11 (двенадцатый столбец слева). Таким образом, соответствующий выход равен $\mathbf{y} = (0,1,0,1)$.

S-блок номер 1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S-блок номер 2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S-блок номер 3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	5	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S-блок номер 4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	14
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	4
S-блок номер 5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S-блок номер 6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S-блок номер 7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S-блок номер 8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Опишем теперь алгоритм выработки цикловых ключей $k_1 - k_{16}$ (см. рис.16).

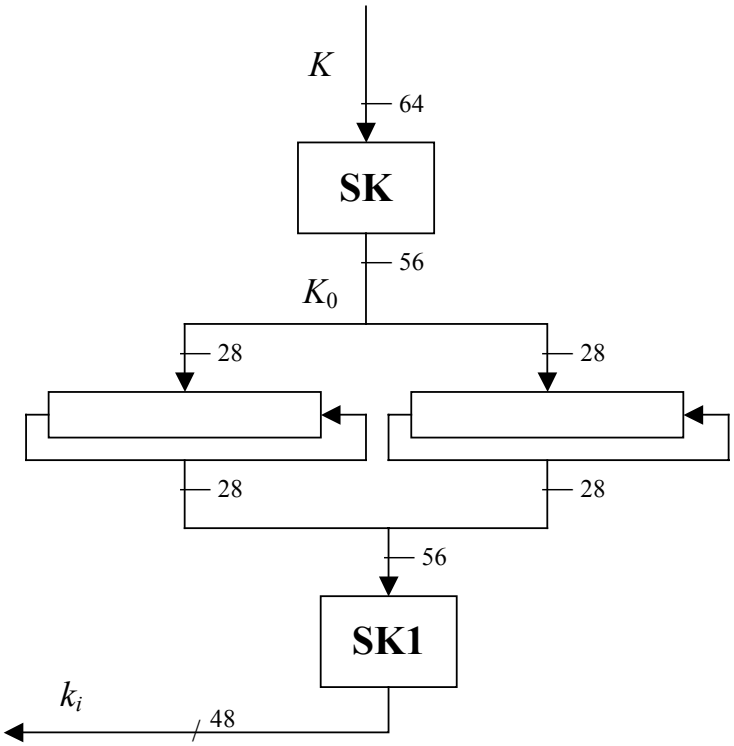


Рис. 16

1. Исходный 64-битный ключ K преобразуется с помощью усеченной перестановки **SK** в 56-битный вектор K_0 (отбрасываются биты 8, 16, 24, 32, 40, 48, 56, 64, которые обычно используются для контроля целостности ключа K).

Усеченная перестановка SK

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

2. 56-битный вектор K_0 записывается в два регистра по 28 бит каждый. Содержимое регистров циклически сдвигается влево на 1 или 2 бита в зависимости от номера цикла шифрования.

Величина сдвига ключевых регистров по циклам шифрования

Номер цикла	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Величина сдвига	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3. По усеченной перестановке **SK1** выбирается 48 бит из 56, которые и составляют цикловой ключ k_i.

Усеченная перестановка SK1

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

Алгоритм расшифрования

Расшифровка текста, осуществляется с помощью того же самого алгоритма, что и шифрование, за исключением того, что если для шифрования использовались цикловые ключи k_1, k_2, ..., k_{16} , то для расшифрования используются ключи k_{16}, k_{15}, ..., k_1. Фактически можно считать, что ключи генерируются по той же схеме, но только циклический сдвиг в регистрах осуществляется вправо, а величина сдвига выбирается из таблицы в обратном порядке. При реализации алгоритма на практике ключи, как правило, генерируются заранее, а не в процессе шифрования или расшифрования.

Режимы работы алгоритма DES

Алгоритм блочного шифрования DES может работать в нескольких ***режимах*** (*modes*), рекомендованных к использованию в публикации NIST [11] от 1980 года. Вообще говоря, эти режимы никак не связаны с конкретными особенностями алгоритма DES и пригодны для работы любой системы блочного шифрования. Для описания этих режимов введем следующие обозначения:

- E_k – шифрующее преобразование, осуществляемое алгоритмом блочного шифрования с ключом k;
- D_k – преобразование расшифрования, осуществляемое алгоритмом блочного шифрования с ключом k;
- M_i – i-й информационный блок открытого текста, $i = 1,2,...$;
- C_i – i-й информационный блок шифртекста, $i = 1,2,...$;
- IV – initialization vector – ***синхропосылка***. Данный вектор также иногда называют маркантом.
- $T^{(m)}(A)$ – информационный блок, образованный первыми m битами информационного блока A;
- $\hat{R}^{(m)}(A,B)$ – информационный блок, получаемый из информационных блоков A и B путем сдвига блока A влево на m разрядов (при этом первые m битов отбрасываются) и помещения на место последних разрядов первых m битов из блока B;
- $\oplus$ – векторное сложение соответствующих информационных блоков.

Открытый текст P последовательно разбивается на информационные блоки M_1, M_2, … . Размер блоков определяется соответствующим режимом пользования. Результатом работы является последовательность информационных блоков C_1, C_2, …, того же размера, совокупность которых образует шифртекст. В этих обозначениях режимы работы DES можно описать следующим образом:

Название режима	Размер информационных блоков	Преобразования, осуществляемые режимом
ECB (*electronic code book*)	$M_i, C_i \in \mathbb{Z}_2^{64}$	$C_i = E_k(M_i); \quad M_i = D_k(C_i).$
CBC (*ciphertext block chaining*)	$M_i, C_i \in \mathbb{Z}_2^{64}$	$C_i = E_k\left(M_i \oplus C_{i-1}\right);$ $M_i = D_k\left(C_i\right) \oplus C_{i-1}; \quad C_0$=IV.
CFB (*ciphertext feedback*)	$M_i, C_i \in \mathbb{Z}_2^m$, $1 \le m \le 64$	$C_i = M_i \oplus \Gamma_i; \quad M_i = C_i \oplus \Gamma_i$ $\begin{cases} \Gamma_i = T^{(m)}\left(E_k(W_i)\right) \\ W_i = \hat{R}^{(m)}\left(W_{i-1}, C_{i-1}\right); \end{cases} \quad W_1$=IV. В случае m=64 работа режима CFB может быть описана проще: $C_i = M_i \oplus E_k(C_{i-1});$ $M_i = C_i \oplus E_k(C_{i-1}); \quad C_0$=IV.
OFB (*output feedback*)	$M_i, C_i \in \mathbb{Z}_2^m$, $1 \le m \le 64$	$C_i = M_i \oplus \Gamma_i; \quad M_i = C_i \oplus \Gamma_i$ $\begin{cases} \Gamma_i = T^{(m)}\left(E_k(W_i)\right) \\ W_i = \hat{R}^{(m)}\left(W_{i-1}, \Gamma_{i-1}\right); \end{cases} \quad W_1$=IV. В случае m=64 работа режима OFB может быть описана проще: $C_i = M_i \oplus \Gamma_i; \quad M_i = C_i \oplus \Gamma_i$ $\Gamma_i = E_k(\Gamma_{i-1}); \quad \Gamma_0$=IV.

ECB (electronic code book) – Электронная кодовая книга
Приведенное выше описание алгоритма соответствует режиму работы называемому ***электронная кодовая книга***. Это название оправдано в силу того, что при работе в таком режиме одинаковые блоки открытого текста переходят в одинаковые блоки шифрованного, т.е. фактически осуществляется простая замена в алфавите мощности 2^{64}. Преимуществом данного метода является произвольность доступа к информации. Недостатком – то, что при наличии в открытом тексте часто повторяющихся фрагментов или стандартов (стандартное начало, стандартное окончание и т.д.) шифрованный текст сохраняет все эти особенности.

CBC (Cipher Block Chaining) – Зацепление блоков шифра
Второй режим специфицированный для DES алгоритма называется режимом ***зацепления блоков шифра***. Смысл его заключается в побитном сложении перед шифрованием очередного блока открытого текста с результатом шифрования предыдущего блока. Первый блок складывается со специальным 64-битным вектором, называемым ***синхропосылкой*** или ***маркантом*** (*initialization vector – IV*). Он, как правило, не является секретным и передается получателю сообщения отправителем в открытом виде. При использовании различных синхропосылок даже одинаковые открытые сообщения будут зашифрованы по разному. Неповторяемость синхропосылок можно достигнуть путем присваивания каждому сообщению своего уникального номера, который и использовать в качестве марканта.

CFB (Cipher Feedback) – Обратная связь по шифрованному тексту
Данный режим позволяет использовать блочный алгоритм в режиме поточного шифрования (см. гл. 9). Также этот режим иногда называют режимом ***гаммирования с самовосстановлением***. Для реализации этого режима криптосхема дополняется проходным двоичным регистром длины 64, начальным состоянием которого является маркант, требования к которому аналогичны требованиям к марканту в режиме CBC. Для того, чтобы подчеркнуть, что открытый текст поступает на шифрование блоками по m бит, такой режим называют m-битным CFB. Теоретически возможно использование значения параметра m от 1 до 64, однако как нетрудно видеть скорость шифрования в режиме 1 битного CFB в 64 раза ниже, чем например, у шифрования в режиме ECB.

OFB (Output Feedback) – Обратная связь по выходу тексту
Этот режим похож на режим CFB, включая требования к марканту. Здесь регистр, содержащий вектор W, в совокупности с алгоритмом шифрования является автономным автоматом. Заполнение этого регистра играет роль внут-

реннего состояния этого автомата. Очередной выход автомата побитно складывается с соответствующим информационным блоком открытого текста. Существует одна особенность использования этого режима: по соображениям безопасности следует использовать только режим 64-битного OFB, несмотря на то что сертифицированы и другие режимы.

Все приведенные выше режимы специфицированы для алгоритма DES и имеют рекомендации по их применению. Так банковский стандарт предписывает использование режимов ECB и CBC для шифрования данных, CBC и 64-битный CFB для аутентификации. При этом режим ECB рекомендуется применять исключительно для шифрования ключей. Каждому из приведенных режимов свойственны свои преимущества и недостатки. Тщательное исследование этих вопросов проводилось в работе [29], где рассматривались различные режимы работы систем блочного шифрования и сравнивалась их устойчивость по отношению к активному навязыванию информации.

Существует еще несколько не специфицированных режимов работы алгоритма DES:

Название режима	Размер информационных блоков	Преобразования, осуществляемые режимом
PBC (*plaintext block chaining*)	$M_i, C_i \in \mathbb{Z}_2^{64}$	$C_i = E_k(M_i) \oplus M_{i-1};$ $M_i = D_k(C_i \oplus M_{i-1});$ M_0=IV.
PFB (*plaintext feedback*)	$M_i, C_i \in \mathbb{Z}_2^{64}$	$C_i = M_i \oplus E_k(M_{i-1});$ $M_i = C_i \oplus E_k(M_{i-1});$ M_0=IV.
CBCPD (*ciphertext block chaining of plaintext difference*) или PCBC (*propagation cipher block chaining*)	$M_i, C_i \in \mathbb{Z}_2^{64}$	$C_i = E_k(M_i \oplus M_{i-1} \oplus C_{i-1});$ $M_i = D_k(C_i) \oplus C_{i-1} \oplus M_{i-1};$ (M_0, C_0)=IV .

Существуют и другие не специфицированные режимы, однако, в силу их практической не распространенности интереса они не представляют.

§ 5. ГОСТ 28147-89: Стандарт СССР и РФ, 1989 г.

Отечественный стандарт шифрования носит официальное название «Алгоритм криптографического преобразования ГОСТ 28147-89» [3]. Как явствует из его номера, стандарт был принят в СССР в 1989 г. Далее будем называть его просто алгоритмом шифрования ГОСТ или просто ***ГОСТ*** для краткости. Если охарактеризовать алгоритм ГОСТ в самом общем виде, то он является блочным шифром, построенным по схеме Фейстеля с 32 циклами шифрования. Длина информационного блока – 64 бита, длина ключа – 256 бит.

Переходя к более подробному описанию алгоритма, отметим, прежде всего, что в отличии от алгоритма DES, в официальной схеме алгоритма ГОСТ формально отсутствуют начальная и финальная перестановки битов информационного блока. Однако, в стандарте ГОСТ указывается, что информационный блок, который в дальнейшем будет преобразовываться алгоритмом, образуется битами соответствующего блока открытого текста, записанными в обратном порядке. Точнее, если блок открытого текста $M = (m_1, m_2, ..., m_{64})$, то полублоки информационного блока имеют вид $\mathbf{l}_0 = (m_{64}, m_{63}, ..., m_{33})$ и $\mathbf{r}_0 = (m_{32}, m_{31}, ..., m_1)$. В свою очередь, если после 32 циклов работы алгоритма значения полублоков $\mathbf{l}_{32} = (c_{64}, c_{63}, ..., c_{33})$, $\mathbf{r}_{32} = (c_{32}, c_{31}, ..., c_1)$, то шифртекст $C = (c_1, c_2, ..., c_{64})$. Поэтому можно считать, что схема алгоритма ГОСТ полностью совпадает со схемой алгоритма DES, при этом используется 32 цикла шифрования, а начальная и финальная подстановки совпадают и задаются строкой 64, 63, …, 2,1.

Далее, аналогично алгоритму DES, в конце последнего цикла шифрования (в ГОСТ это 32-й цикл) левый и правый полублоки информационного блока не переставляются. Причина этого такая же, как и для DES: в этом случае алгоритм расшифрования структурно совпадает с алгоритмом шифрования. Единственное отличие – обратный порядок использования цикловых ключей.

Основные отличия алгоритма ГОСТ от алгоритма DES – в строении функции $\boldsymbol{f}$ и алгоритме выработки цикловых ключей. И в том и в другом случае преобразования, используемые в алгоритме ГОСТ, проще для программной реализации.

Функция $\boldsymbol{f}$ осуществляет отображение $\mathbb{Z}_2^{32} \times \mathbb{Z}_2^{32} \to \mathbb{Z}_2^{32}$ (см. рис.17). Это отображение образовано следующими преобразованиями:

1. 32-битный вектор данных, составляющих полублок $\mathbf{r}_{i-1}$, и 32-битный цикловой ключ k_i рассматриваются как 32-разрядные двоичные числа и складывается по модулю 2^{32}.
2. 32-битный вектор, полученный в п.1, делится на восемь 4-битных слов. Каждое из них поступает на вход соответствующего $S_{4\times4}$-блока, которые обозначены как $\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_3}$, $\mathbf{S_4}$, $\mathbf{S_5}$, $\mathbf{S_6}$, $\mathbf{S_7}$, $\mathbf{S_8}$. Информация об S-блоках будет приведена ниже. Восемь 4-битных слов, являющихся выходами S-блоков, конкатенируются в 32-битный вектор.
3. 32-битный вектор, полученный в п.2, поступает на вход перестановки **P**, (P-блок размера 32×32), которая осуществляет циклический сдвиг влево на 11 позиций.

Полученный на выходе перестановки **P** информационный блок является значением $\boldsymbol{f}(\mathbf{r}_{i-1}, k_i)$.

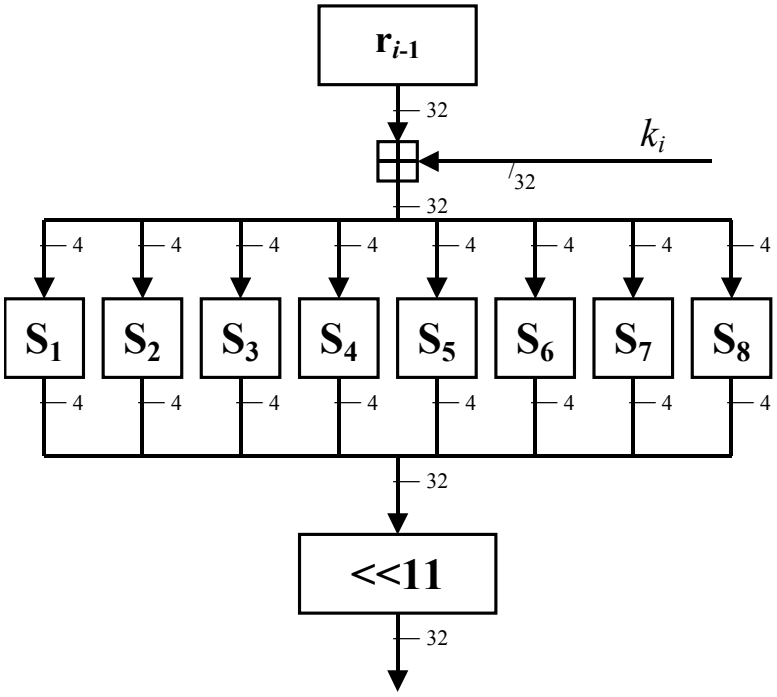


Рис.17

В отличии от алгоритма DES, S-блоки $\mathbf{S_1} - \mathbf{S_8}$ алгоритма ГОСТ в стандарте не задаются. Стандарт лишь определяет, что набор S-блоков (называемых в

стандарте узлами замены) является долговременным ключевым элементом, общим для сети ЭВМ. Однако, как было показано в ряде работ, выбор в качестве S-блоков произвольных подстановок может привести к снижению криптостойкости алгоритма. Так что разработка критериев, позволяющих получать «хорошие» S-блоки, является важной задачей как теоретической, так и практической криптографии. В описаниях алгоритма ГОСТ, встречающихся в открытой литературе, обычно приводятся S-блоки, используемые в реализации для Центрального Банка РФ [7]:

S-блоки:

S-блок номер 1															
4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3

S-блок номер 2															
14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9

S-блок номер 3															
5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11

S-блок номер 4															
7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3

S-блок номер 5															
6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2

S-блок номер 6															
4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14

S-блок номер 7															
13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12

S-блок номер 8															
1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Опишем теперь алгоритм выработки цикловых ключей $k_1 - k_{32}$.

Алгоритм выработки цикловых ключей.

Общий 256-битный ключ алгоритма $K = (x_1, x_2, ..., x_{256})$ записывается в восемь 32-разрядных регистров X_0, X_1, …, X_7:

$X_0 = (x_{32}, x_{32}, ..., x_1)$, $X_1 = (x_{64}, x_{63}, ..., x_{33})$, …, $X_7 = (x_{256}, x_{255}, ..., x_{225})$.

В качестве цикловых ключей $k_1, k_2, ..., k_{32}$ выбирается содержание соответствующего регистра по правилу:

$$k_1 = X_0, \quad k_2 = X_1, \ ..., \quad k_8 = X_7,$$
$$k_9 = X_0, \ k_{10} = X_1, \ ..., \ k_{16} = X_7,$$
$$k_{17} = X_0, \ k_{18} = X_1, \ ..., \ k_{24} = X_7,$$
$$k_{25} = X_7, \ k_{26} = X_6, \ ..., \ k_{32} = X_0.$$

Расшифрование.

Расшифрование осуществляется тем же алгоритмом, но порядок использования цикловых ключей меняется на обратный

Режимы работы алгоритма ГОСТ.

В стандарте специфицированы следующие режимы работы алгоритма:

1. Простая замена
2. Гаммирование
3. Гаммирование с обратной связью
4. Выработка имитовставки

Опишем эти режимы, используя те же обозначения, что были использованы в предыдущем разделе при описании режимов работы алгоритма DES.

Режим простой замены. Данный режим уже описан выше как режим ECB и в комментариях не нуждается.

Режим гаммирования. Данный режим несколько отличается от описанного выше режима OFB, выглядит он следующим образом.

Начальное состояние регистров A и B является маркантом, который передается вместе с шифрованным текстом. Данное состояние зашифровывается в режиме простой замены и помещается в регистры A и B. После этого начинается выработка гаммы блоками по 64 бита по следующему правилу:

- Константа C_1 складывается с содержимым регистра A по правилу
$$A = A + C_1 - 1 (\mathrm{mod}\ 2^{32} - 1) + 1.$$
Константа C_2 прибавляется по модулю 2^{32} к содержимому регистра B.
- Регистры A и B образуют начальный информационный блок шифрующего алгоритма. Соответствующий этому блоку шифртекст рассматривается как очередные 64 бита гаммы. После чего алгоритм возвращается к выполнению предыдущего пункта.

Константы C_1 и C_2 в шестнадцатеричном представлении имеют вид
$$C_1 = 01010104h, \quad C_2 = 01010101h.$$

Аналогично режиму OFB, регистр, содержащий вектор (A, B), в совокупности с алгоритмом шифрования является автономным автоматом. Заполнение регистра играет роль внутреннего состояния автомата. Однако, в отличии от режима OFB, очередное состояние автомата определяется с помощью указанного выше суммирования с константами. Шифрующий алгоритм в вычислении очередного внутреннего состояния не участвует. Такой автомат называется ***автоматом счетчикового типа***.

Режим гаммирования с обратной связью.

Данный режим является стандартным режимом CFB со значением параметра m=64.

Режим выработки имитовставки.

Данный режим предназначен для выработки ***имитовставки*** – специальной контрольной суммы, зависящей от ключа и предназначенной для контроля целостности блока переданных данных. Для этого используется алгоритм ГОСТ не с 32, а с 16 итерациями, соответствующими первым 16 циклам шифрования, описанным для режима простой замены. Первый блок открытого текста шифруется 16 циклами шифрования алгоритма ГОСТ. Затем результат побитно складывается со вторым блоком открытого текста, после чего процедура шифрования повторяется. Имитовставка образуется битами, выбранными из информационного блока, полученного после обработки всего открытого текста. После расшифрования проводится аналогичная процедура и полученное значение сравнивается с переданной имитовставкой. При их совпадении считается, что имело место корректная передача и правильное расшифрование полученного шифртекста.

Название режима	Размер информационных блоков	Преобразования, осуществляемые режимом
Режим простой замены	$M_i, C_i \in \mathbb{Z}_2^{64}$	Полностью совпадает с режимом ECB (*electronic code book*)
Режим гаммирования	$M_i, C_i \in \mathbb{Z}_2^{64}$	$C_i = M_i \oplus \Gamma_i; \quad M_i = C_i \oplus \Gamma_i$ $\begin{cases} \Gamma_i = E_k(A_i, B_i); \quad A_i, B_i \in \mathbb{Z}_2^{32} \\ A_i = A_{i-1} + C_1 - 1 (\mathrm{mod}\ 2^{32} - 1) + 1, \\ B_i = B_{i-1} + C_2 (\mathrm{mod}\ 2^{32}); \\ (A_0, B_0) = E_k(\mathrm{IV}). \end{cases}$
Режим гаммирования с обратной связью	$M_i, C_i \in \mathbb{Z}_2^{64}$,	Полностью совпадает с режимом CFB (*ciphertext feedback*) со значением параметра m=64.
Режим выработки имитовставки	$M_i, I_i \in \mathbb{Z}_2^{64}$,	$I_i = E_k^{(16)}\left(M_i \oplus I_{i-1}\right); \quad I_0 = 0.$

§ 6. AES (Rijndael): Advanced Encryption Standard – стандарт США, 2001 г.

В 1997 г. Национальный Институт Стандартов и Технологий США (NIST)[1)] объявил о начале конкурса на новый стандарт криптографической защиты данных – ***AES*** (Advanced Encryption Standard). Хотя существующий с 1977 г. криптоалгоритм DES и остается действующим стандартом, он считается устаревшим по многим параметрам, среди которых малая длина ключа[2)], неудобство реализации на современных процессорах, малое быстродействие. Несомненным достоинством алгоритма DES является его стойкость. За 25 лет интенсивного криптоанализа не было найдено методов вскрытия этого шифра, существенно отличающихся по эффективности от полного перебора всех ключей ключевого пространства. К новому стандарту были предъявлены следующие требования:

- криптоалгоритм должен быть симметричным блочным шифром, допускающим размеры ключей в 128, 192 или 256 бит;
- криптоалгоритм должен быть удобен как для аппаратной, так и для программной реализации;
- криптоалгоритм должен удовлетворять современным требованиям по следующим параметрам: стойкость, скорость, стоимость, гибкость.

В конце 2000 г. победителем конкурса был объявлен криптоалгоритм Rijndael. С мая 2002 г. этот алгоритм с небольшими модификациями о которых будет сказано ниже, стал официальным стандартом шифрования данных AES [8]. Материалы конкурса AES, в том числе подробную документацию по алгоритмам – финалистам конкурса, а также результаты их анализа можно найти на Web-сайте [http://www.nist.gov/encryption/aes/].

[1)] Прежнее название – Национальное Бюро Стандартов.

[2)] DES остается стандартом лишь в качестве составной части алгоритма TDEA (Triple DES) со 112- или 168-битным ключом.

Структура шифра

Общие сведения

AES – это итерационный блочный шифр, имеющий следующие параметры:

- длина информационного блока – 128 бит[2)];
- длина ключа – варьируется и может равняться 128, 192 или 256 битам;
- число циклов шифрования зависит от длины ключа и равняется 10, 12 или 14 циклам.

Промежуточные результаты преобразований, выполняемых над информационным блоком в рамках криптоалгоритма, называются в дальнейшем ***состояниями*** (*state*). Состояние рассматривается в виде прямоугольного (4×4)-массива байтов, по 4 байта в столбце. Цикловые ключи также представляются в виде (4×4)-массива байтов. На рис.18 изображены состояние и цикловой ключ и указывается нумерация байтов в соответствующих массивах.

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}

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. 18

128-битный блок входных данных представляется в виде 16 байтов состояния в следующем порядке: s_{00}, s_{10}, s_{20}, s_{30}, s_{01}, s_{11}, s_{21}, s_{31}, ... После завершения

[2)] Основное отличие алгоритма Rijndael от принятого стандарта AES состоит в том, что алгоритм Rijndael допускает также информационные блоки длиной в 192 или 256 бит. Алгоритм выработки цикловых ключей и число циклов шифрования в Rijndael изменяется в зависимости от выбранной длины информационного блока.

процесса шифрования выходные данные получаются из байтов состояния в том же порядке.
Зависимость числа циклов шифрования от длины ключа показана в таблице:

Длина ключа	128	192	256
Число циклов шифрования	10	12	14

Табл. 3

Цикловое преобразование

Любой цикл шифрования, кроме последнего, состоит из четырех различных преобразований. На псевдо-Си это выглядит следующим образом:

```
Round (State, RoundKey)
{
ByteSub(State); // замена байтов
ShiftRow(State); // сдвиг строк
MixColumn(State); // перемешивание столбцов
AddRoundKey(State, RoundKey); // сложение с цикловым ключом
}
```

Последний цикл шифрования немного отличается от остальных[1)]. Вот как он выглядит:

```
FinalRound(State, RoundKey)
{
ByteSub(State); // замена байтов
ShiftRow(State); // сдвиг строк
AddRoundKey(State, RoundKey); // сложение с цикловым ключом
}
```

Как можно заметить, последний цикл отличается от предыдущих циклов только отсутствием перемешивания столбцов. Каждое из приведенных преобразований определено далее.

[1)] Очевидно, что подобный выбор последнего цикла шифрования мотивирован стремлением придать алгоритму расшифрования структуру, тождественно совпадающую со структурой алгоритма шифрования.

Замена байтов (ByteSub).

Преобразование ByteSub представляет собой нелинейную замену байтов, выполняемую независимо для каждого байта состояния. Можно считать, что преобразование *ByteSub* действует на каждый байт состояния, преобразуя его с помощью одного и того же обратимого S-блока размера 8×8 (см. рис.19).

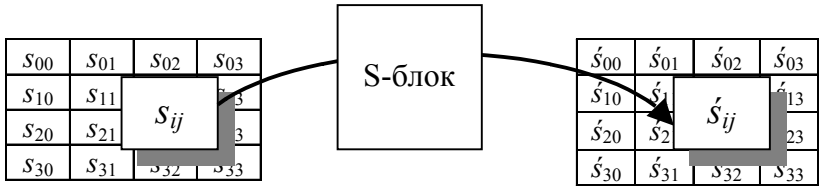


Рис. 19

Таблица замены для данного S-блока является инвертируемой и построена как композиция двух преобразований:

- переход к обратному элементу относительно умножения в поле $GF(2^8)$, при этом нулевой элемент $00h$ переходит сам в себя;
- аффинное преобразование 8-мерного двоичного вектора, полученного в п.1:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Это аффинное преобразования может быть также задано в виде системы уравнений:

$$y_0 = x_0 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus 1 \qquad y_4 = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4$$
$$y_1 = x_0 \oplus x_1 \oplus x_5 \oplus x_6 \oplus x_7 \oplus 1 \qquad y_5 = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus 1$$
$$y_2 = x_0 \oplus x_1 \oplus x_2 \oplus x_6 \oplus x_7 \qquad y_6 = x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus 1$$
$$y_3 = x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_7 \qquad y_7 = x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7$$

Применение описанного выше преобразования (S-блока) ко всем байтам состояния обозначено как ByteSub(State). Операция обратная к ByteSub – это замена байтов с использованием инвертированной таблицы. Обратимость операции ByteSub следует из обратимости аффинного преобразования и наличия у каждого ненулевого элемента поля $GF(2^8)$ обратного элемента.

Сдвиг строк (ShiftRow)

Строки матрицы состояния (кроме нулевой строки) циклически сдвигаются влево на различное число байт. Строка 1 сдвигается на 1 байт, строка 2 – на 2 байта и строка 3 – на 3 байта. Операция сдвига строк состояния на указанные величины обозначена как ShiftRow(State). На рис.20 показано влияние преобразования ShiftRow на состояние State.

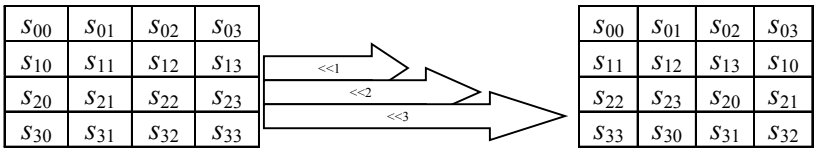


Рис. 20

Перемешивание столбцов (MixColumn)

В этом преобразовании столбцы состояния рассматриваются как многочлены над полем GF(2^8) и умножаются по модулю $M(x) = x^4 + 1$ на многочлен $c(x) = (03h)x^3 + (01h)x^2 + (01h)x + (02)$. Коэффициенты этого многочлена являются элементами поля GF(2^8) и приведены в шестнадцатеричной записи: (03h) = (00000011). Многочлен $c(x)$ взаимно прост с многочленом $M(x)$ и, следовательно, операция умножения на $c(x)$ по модулю $M(x)$ обратима.

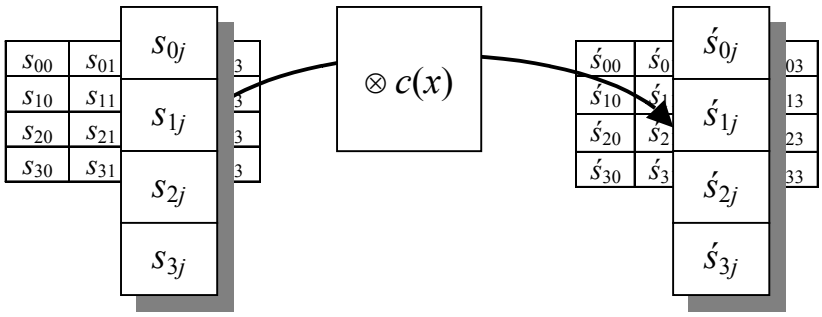


Рис. 21

Нетрудно заметить, что это преобразование является линейным для векторов из $\left(GF(2^8)\right)^4$ и может быть также представлено в матричном виде следующим образом

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{pmatrix} 02h & 03h & 01h & 01h \\ 01h & 02h & 03h & 01h \\ 01h & 01h & 02h & 03h \\ 03h & 01h & 01h & 02h \end{pmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}.$$

Применение этой операции ко всем четырем столбцам состояния обозначено как MixColumn(State). Для обращения операции MixColumn необходимо умножить столбцы преобразованного состояния на многочлен обратный к многочлену $c(x)$ по модулю x^4+1, т.е. на многочлен $d(x)=(0\mathrm{B}h)x^3+(0\mathrm{D}h)x^2+(09h)x+(0\mathrm{E}h)$ (нетрудно проверить, что в кольце многочленов над полем $\mathrm{GF}(2^8)$ произведение $c(x)\otimes d(x)=1 \bmod (x^4+1)$).

Сложение с цикловым ключом (AddRoundKey).

В данной операции цикловой ключ, представленный в виде матрицы байтов, складывается с матрицей состояния посредством простого поразрядного XOR (рис.22). Эта операция обозначается как AddRoundKey(State, RoundKey) и, очевидным образом, обратима.

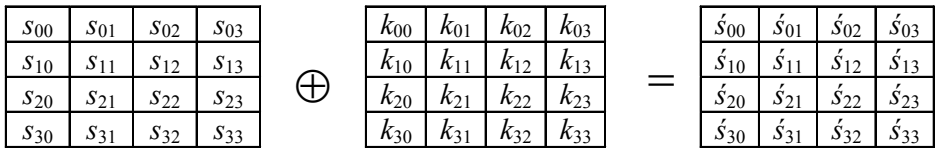


Рис. 22

Цикловой ключ вырабатывается из ключа шифрования посредством алгоритма выработки цикловых ключей (key schedule). Длина циклового ключа должна быть равна длине информационного блока.

Процесс шифрования для алгоритма AES можно кратко проиллюстрировать схемами, изображенными на рис.23: общая схема алгоритма (слева), схема одного цикла шифрования (посередине) и схема заключительного цикла шифрования (справа)

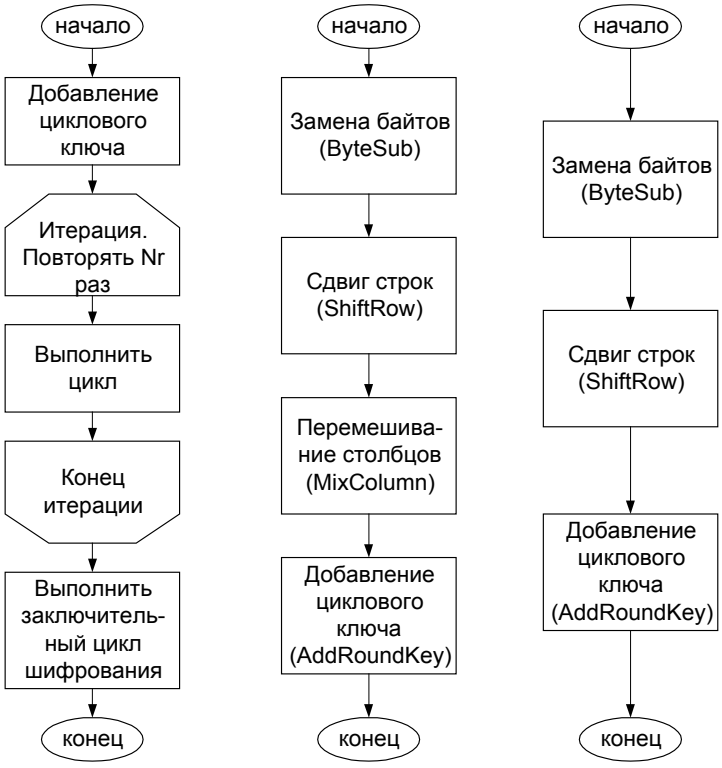


Рис. 23

Алгоритм выработки ключей (Key Schedule)

Цикловые ключи получаются из ключа шифрования с помощью алгоритма выработки ключей. Он состоит из двух частей: расширение ключа (Key Expansion) и выбор циклового ключа (Round Key Selection).

Основные принципы алгоритма выработки ключей выглядят следующим образом:

- общее число бит цикловых ключей равно длине блока, умноженной на число циклов шифрования плюс 1 (например, при длине блока в 128 бит и 10 циклах шифрования требуется 1408 бит цикловых ключей);
- ключ шифрования расширяется в ***расширенный ключ*** (*Expanded Key*); расширенный ключ представляет собой линейный массив 4-байтовых слов;
- цикловые ключи берутся из расширенного ключа следующим образом: первый цикловой ключ содержит первые 4 слова, второй – следующие 4 слова и т.д.

Расширение ключа (Key Expansion)

Исходный ключ шифрования состоит из N_k 4-байтовых слов, где N_k = 4, 6, или 8. Расширенный ключ представляет собой линейный массив 4-байтовых слов и обозначен как $W\left[4\left(N_r+1\right)\right]$, где N_r – число циклов шифрования.

Первые N_k слов содержат ключ шифрования. Все остальные слова определяются рекурсивно из слов с меньшими индексами. Алгоритм выработки ключей зависит от величины N_k. Ниже приведена версия для N_k равного 4 или 6 и версия для N_k = 8.

Для $N_k \le 6$ имеем:

```
KeyExpansion(CipherKey,W)
{
for (i = 0; i < Nk; i++) W[i] = CipherKey[i];
for (j = Nk; j < Nb*(Nk+1); j+=Nk)
{
W[j] = W[j-Nk] ^ SubByte( Rotl( W[j-1] ) ) ^ Rcon[j/Nk];
for (i = 1; i < Nk && i+j < Nb*(Nr+1); i++)
W[i+j] = W[i+j-Nk] ^ W[i+j-1];
}
}
```

Как можно заметить, первые N_k слов заполняются ключом шифрования. Каждое последующее слово *W*[*i*] получается посредством XOR предыдущего слова *W*[*i*–1] и слова $W\left[i-N_k\right]$. находящегося на N_k позиций левее. Для слов, позиция которых кратна N_k, перед операцией XOR применяется преобразование к слову *W*[*i*–1], а затем прибавляется цикловая константа. Преобразование содержит циклический сдвиг байтов в слове, обозначенный как Rotl, затем следует SubByte – замена байт, описанная выше. Для $N_k > 6$ имеем:

```
KeyExpansion(CipherKey,W)
{
for (i=0; i<Nk; i++) W[i]=CipherKey[i];
for (j=Nk; j<Nb*(Nk+1); j+=Nk)
{
W[j] = W[j-Nk] ^ SubByte(Rotl(W[j-1])) ^ Rcon[j/Nk];
for (i=1; i<4; i++) W[i+j] = W[i+j-Nk] ^ W[i+j-1];
W[j+4] = W[j+4-Nk] ^ SubByte(W[j+3]);
for (i=5; i<Nk; i++) W[i+j] = W[i+j-Nk] ^ W[i+j-1];
}
}
```

Отличие по сравнению с ранее рассмотренной схемой состоит в применении преобразования SubByte для каждого 4-го слова массива *W*.
Цикловая константа не зависит от N_k и определяется следующим образом:

Rcon[i] = (RC[i], '00' , '00' , '00'), где
RC[0]='01'
RC[i]=xtime(Rcon[i-1])

Процедуру расширения ключа можно проиллюстрировать схемами, изображенными на рис.24.

Выбор циклового ключа

i-й цикловой ключ получается из слов массива циклового ключа от $W\left[4i+1\right]$ и до $W\left[4\left(i+1\right)\right]$. Алгоритм выработки ключей можно реализовывать и без использования массива $W\left[4\left(N_r+1\right)\right]$**.** Для реализаций, в которых существенно требование к занимаемой памяти, цикловые ключи могут вычисляться на лету посредством использования буфера длиной N_k слов.

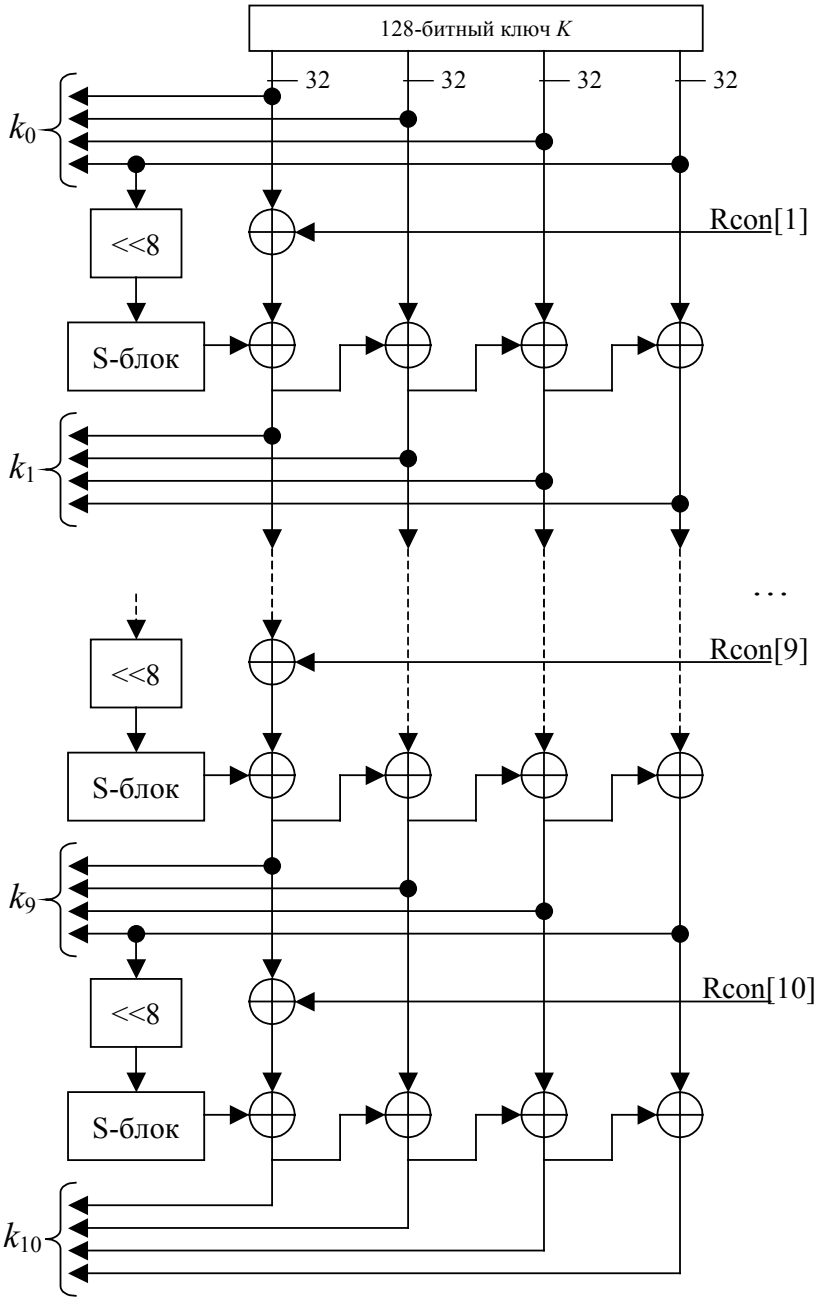

128-битный ключ K
32
32
32
32
k0
<<8
Rcon[1]
S-блок
k1
...
<<8
Rcon[9]
S-блок
k9
<<8
Rcon[10]
S-блок
k10

Рис. 24

Процедура шифрования

В качестве входных данных имеем открытый текст. Выходные данные – зашифрованный текст. Для шифрования открытого текста выполняем следующие действия:

- разбиваем текст на блоки (текст в блок записывается по столбцам);
- шифруем каждый блок.

Шифрование блока состоит из следующих этапов:

- начального сложения с цикловым ключом;
- $N_r - 1$ циклов шифрования;
- заключительного цикла шифрования.

На псевдо-Си это выглядит следующим образом:

```
Rijndael (State, CipherKey)
{
KeyExpansion(CipherKey, ExpandedKey); // расширение ключа
AddRoundKey(State, ExpandedKey); // сложение с цикловым ключом
For ( i=1 ; i<Nr ; i++) Round(State,ExpandedKey+Nb*i); // циклы шифрования
FinalRound(State, ExpandedKey+Nb*Nr); // заключительный цикл
}
```

Расширенный ключ должен всегда получаться из ключа шифрования и никогда не указывается напрямую. На выбор ключа шифрования никаких ограничений не накладывается.

Процедура расшифрования

Нетрудно показать, что все преобразования, используемые в алгоритме AES – обратимы. Более того, свойства используемых преобразований позволяет получить структуру обратного алгоритма тождественно совпадающую со структурой прямого алгоритма. Естественно, что в обратном алгоритме порядок использования цикловых ключей должен быть обратным. Для расшифрования текста необходимо разбить текст на блоки и к каждому блоку применить обратный алгоритм.

Задачи

1. Доказать, что для любой функции $f : \mathbb{Z}_2^m \times \mathbb{Z}_2^k \to \mathbb{Z}_2^m$, m>1, преобразование Фейстеля $F(l, r) = (r, l \oplus f(r))$ осуществляет четную подстановку на множестве $\mathbb{Z}_2^{2m}$.

2. В чем смысл отсутствия перестановки левого и правого полублоков информационного блока в конце последнего цикла шифрования у алгоритмов DES и ГОСТ ?

3. Сколько должно быть циклов шифрования в шифре Фейстеля без перестановки полублоков (рис. 6), чтобы одна и та же схема вычислений реализовывала и алгоритм шифрования и алгоритм расшифрования?

4. Пусть шифр Фейстеля без перестановки полублоков (рис. 6) таков, что алгоритм шифрования и алгоритм расшифрования не может быть реализован одной и той же схемой вычислений. Как видоизменить последний цикл шифрования, чтобы такая возможность появилась?

5. Доказать, что обобщенные преобразования Фейстеля реализуют подстановки на соответствующих множествах двоичных векторов.

6. Доказать ***свойство дополнительности*** преобразования, реализуемого шифром DES: если $c = DES(p,k)$, то $DES(\overline{p},\overline{k}) = \overline{c}$, где черта сверху обозначает поразрядное дополнение соответствующего двоичного вектора, т.е. $\overline{\mathbf{x}} = (\overline{x}_1,...,\overline{x}_n) = (x_1 \oplus 1,...,x_n \oplus 1)$. Вывести отсюда, что при использовании специально подобранных пар открытый текст/шифртекст объем ключевого пространства для метода тотального опробования уменьшается с 2^{56} до 2^{55}.

7. Считая S-блоки алгоритма ГОСТ долговременными ключами, оценить общую мощность ключевого пространства алгоритма.

8. Для каждого из описанных в этой главе режимов пользования построить схему на регистрах, реализующую этот режим.

9. Для каждого из описанных в этой главе режимов пользования определить:

- Что будет, если в одном информационных блоке шифртекста будут изменены символы?
- Что будет, если в одном информационных блоке шифртекста выпадут несколько символов?
- Что будет, если в шифртексте выпадет один или несколько информационных блоков?

ЛИТЕРАТУРА

1. Брикелл Э.Ф., Одлижко Э.М. *Криптоанализ: Обзор новейших результатов*. ТИИЭР, т.76, №5 (1988), 75-93.
2. Варфоломеев А.А., Жуков А.Е., Мельников А.Б., Устюжанин Д.Д. *Блочные криптосистемы. Основные свойства и методы анализа стойкости*. М.: МИФИ, 1998
3. *ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования*. М.: 1989
4. Месси Дж.Л. *Введение в современную криптологию*. ТИИЭР, т.76, №5 (1988), с.24-42.
5. Смид М.Э., Бранстед Д.К. *Стандарт шифрования данных: Прошлое и будущее*. ТИИЭР, т.76, №5 (1988), с.43-53.
6. Шеннон К.Э. *Теория связи в секретных системах*. В кн.: Шеннон К.Э. Работы по теории информации и кибернетике. М.: И.Л., 1963.
7. Шнайер Б. *Прикладная криптография*. М.: ТРИУМФ, 2002
8. *Advanced Encryption Standard (AES)*. Federal Information Processing Standard (FIPS) Publication 197, National Institute of Standards and Technology (NIST), US Department of Commerce, Washington D.C., November, 2001.
9. Biham E., Shamir A. *Differential cryptanalysis of DES-like cryptosystems*. Proc. Crypto-90, Lect. Notes Comput. Sci., v.537 (1991), p.2-21.
10. *Data Encryption Standard (DES)*. Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, US Department of Commerce, Washington D.C., January 1977.
11. *DES Modes of Operation.* Federal Information Processing Standard (FIPS) Publication 81, National Institute of Standards and Technology (NIST), US Department of Commerce, Washington D.C., December 1980.
12. Even S., Goldreich O. *On the power of cascade ciphers*. Proc. Crypto-83, D. Chaum (ed.), Plenum Press, N.Y., 1984, p.43-50.
13. Feistel H. *Cryptography and Computer Privacy*. Scientific American, v.228, N 5 (1973), pp. 15-23.
14. Feistel H., Notz W.A., and Smith J.L. *Some cryptographic techniques for machine to machine data communications*. Proc. IEEE, v.63, N 11 (1975), pp.1545-1554.
15. Kaliski B.S., Rivest R.L., Sherman A.T. *Is Data Encryption Standard a group?* Proc. Eurocrypt-85, Lect. Notes Comput. Sci., v.219 (1986), p.81-95.
16. Kaliski B.S.Jr., Rivest R.L., Sherman A.T. *Is DES a pure cipher? (Results of more cycling experiments on DES)*. Proc.Crypto-85, Lect. Notes Comput. Sci., v.218 (1986), p.212-226.
17. Konheim A.G. *Cryptography. A premier*. J.Willey & Sons, N.Y., 1981.

18. Lai X., Massey J.L., Murphy S. *Markov ciphers and differential cryptanalisis.* Proc. Eurocrypt-91, Lect. Notes Comput. Sci., v.547 (1991), p.17-38.
19. Luby M., Rackoff C. *How to construct pseudorandom permutations from pseudorandom functions.* SIAM J. Comput., v.17, N2 (1988), p.373-386.
20. Menezes A., van Oorschot P., and Vanstone S. *Handbook of Applied Cryp tography*, CRC Press, New York, 1997.
21. Meyer C.H., Matyas S.M. *Cryptogrtaphy: a new dimension in computer data security.* J.Willey & Sons, N.Y., 1982.
22. Moore T.E., Tavares S.E. *A layered approach to the design of private key cryptosystems.* Proc.Crypto-85, Lect. Notes Comput. Sci., v.218 (1986), p.227-245.
23. Pieprzyk J. *How to costruct pseudorandom permutations from single pseudo-random function.* Proc. Eurocrypt-90, Lect. Notes Comput. Sci., v.473 (1991), p.140-150.
24. Pieprzyk J., Sadeghiyan B. *Design of hashing algorithms.* Springer-Verlag, Berlin-Heidelberg, 1993, Lect. Notes Comput. Sci., v.756.
25. Schnorr C.P. *On the construction of random number generators and random function generators.* Proc. Eurocrypt-88, Lect. Notes Comput. Sci., v.330 (1988), p.225-232.
26. Zheng Y., Matsumoto T., Hideki I. *Impossibility and optimality results on constructing pseudorandom permutations.* Advances in Cryptology. Proc. Eurocrypt-89, Lect. Notes Comput. Sci., v.434 (1990), pp.412-422.
27. Zheng Y., Matsumoto T., Imai I. *On the construction of block ciphers provably secure and not relying on any unproved hypotheses.* Proc. Crypto-89, Lect. Notes Comput. Sci.
28. Price W.L. *Standards for data security - a change of direction.* Proc. Crypto-87, Lect. Notes Comput. Sci.- 1988, p.3-8
29. Jansen C.J.A., Boekee D.E. *Modes of blockcipher algorithms and their protection against active eavesdropping.* Proc. Eurocrypt-87, Lect. Notes Comput. Sci.- 1988, p.281-286