Лекция 3.
БЛОЧНЫЕ ШИФРЫ
СЕТИ ФЕЙШТЕЛЯ

1. **Общие сведения о блочных шифрах**
2. **Сеть Фейштеля**
3. **Режимы шифрования**

Блочный шифр

разновидность симметричного шифра, оперирующего группами бит фиксированной длины — блоками, размер которых меняется в пределах 64 — 256 бит.

Особенность: в ходе своей работы блочные шифры производят преобразование блока входной информации фиксированной длины и получают результирующий блок того же объема, но недоступный для прочтения сторонним лицам, не владеющим ключом.

Схему работы блочного шифра можно описать функциями

Z=EnCrypt(X,Key) - шифрование

X=DeCrypt(Z,Key) – дешифрование

Ключ Key - параметр блочного криптоалгоритма, представляет собой некоторый блок двоичной информации фиксированного размера.

Исходный (X) и зашифрованный (Z) блоки данных также имеют фиксированную разрядность, равную между собой, но необязательно равную длине ключа.

Принципы построения блочных шифров

1. ***Перемешивание*** - маскирует взаимосвязи между открытым текстом, шифртекстом и ключом (подстановка)
2. ***Рассеивание*** - распространяет влияние отдельных битов открытого текста на возможно больший объем шифртекста (перестановка)
3. ***Итерирование*** - заключается в многократной, состоящей из нескольких циклов обработке одного блока открытого текста. На каждом цикле данные подвергаются специальному преобразованию при участии вспомогательного ключа, полученного из заданного секретного ключа. Выбор числа циклов определяется требованиями криптостойкости и эффективности реализации шифра. Как правило, чем больше циклов, тем выше криптостойкость и ниже эффективность реализации.
4. Практически все современные блочные шифры являются композиционными - т.е состоят из композиции простых преобразований или F=F1oF2oF3oF4o..oFn, где F - преобразование шифра, Fi - простое преобразование, называемое также i-ым циклом шифрования.

Примеры блочных алгоритмов

Название алгоритма	Автор	Размер блока	Длина ключа
IDEA	X. Lia and J. Massey	64 бита	128 бит
CAST128	C. Adans and S. Tavares	64 бита	128 бит
BlowFish	Bruce Schneier	64 бита	128 - 448 бит
ГОСТ	НИИ ***	64 бита	256 бит
TwoFish	Bruce Schneier	128 бита	128 - 256 бит
MARS	Корпорация IBM	128 бита	128 - 1024 бит

- По теории вероятности искомый ключ будет найден с вероятностью ½ после перебора половины всех ключей, на взлом стойкого криптоалгоритма с ключом длины N потребуется в среднем 2^{N-1} проверок.
- стойкость блочного шифра зависит от длины ключа и возрастает экспоненциально с ее ростом. Если на проверку 1 ключа требуется 1 такт, то на взлом 128 битного ключа не менее 10^{21}.

Требования к блочным шифрам

- Функция EnCrypt должна быть обратимой.
- Не должно существовать иных методов прочтения сообщения X по известному блоку Z, кроме как полным перебором ключей Key.
- Не должно существовать иных методов определения, каким ключом Key было произведено преобразование известного сообщения X в сообщение Z, кроме как полным перебором ключей

- Все действия, производимые над данными блочным криптоалгоритмом, основаны на том факте, что преобразуемый блок может быть представлен в виде целого неотрицательного числа из диапазона, соответствующего его разрядности.
- 32-битный блок данных можно интерпретировать как число из диапазона 0..4'294'967'295.
- блок, разрядность которого является «степенью двойки», можно трактовать как несколько независимых неотрицательных чисел из меньшего диапазона (32-битный блок можно представить в виде 2 независимых чисел из диапазона 0..65535 или в виде 4 независимых чисел из диапазона 0..255).

Над этими числами блочным криптоалгоритмом и производятся по определенной схеме следующие действия :

Биективные математические функции		
	Сложение	X'=X+V
	Исключающее ИЛИ	X'=X XOR V
	Умножение по модулю 2^N+1	X'=(X*V) mod (2^N+1)
	Умножение по модулю 2^N	X'=(X*V) mod (2^N)
Битовые сдвиги		
	Арифметический сдвиг влево	X'=X SHL V
	Арифметический сдвиг вправо	X'=X SHR V
	Циклический сдвиг влево	X'=X ROL V
	Циклический сдвиг вправо	X'=X ROR V
Табличные подстановки		
	S-box с (блок подстановки)	X'=S[X,V]
	P-box (блок перестановки)	X'=P[X,V]

В качестве параметра V для любого из этих преобразований может использоваться:

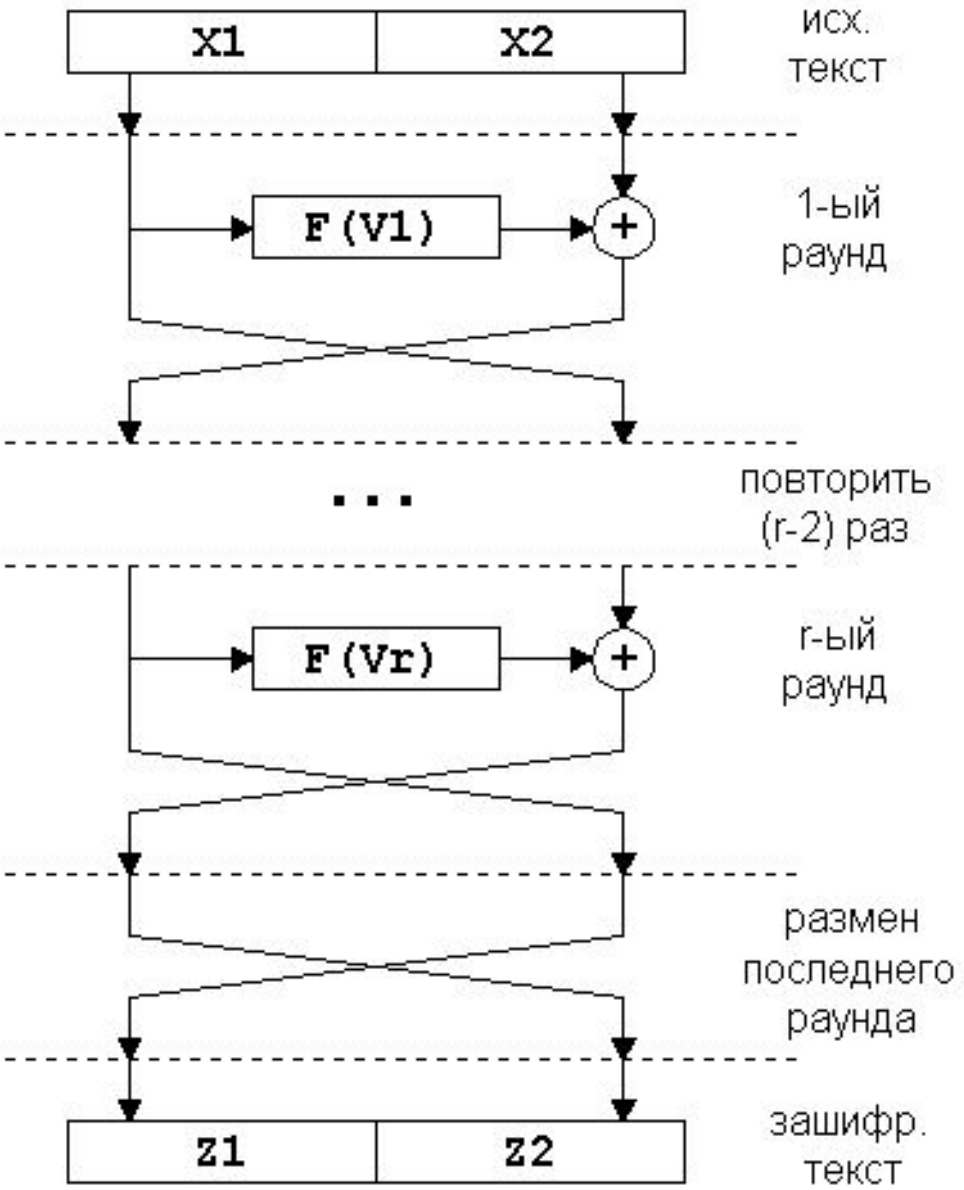
- фиксированное число (например, X'=X+125)
- число, получаемое из ключа (например, X'=X+F(Key))
- число, получаемое из независимой части блока (например, X2'=X2+F(X1))

- Последовательность выполняемых над блоком операций, комбинации вариантов V и сами функции F и составляют «ноу-хау» каждого конкретного блочного криптоалгоритма.

- Характерным признаком блочных алгоритмов является многократное и косвенное использование материала ключа. Для решения этой задачи чаще всего используется не само значение ключа или его части, а некоторая, иногда необратимая функция от материала ключа. В подобных преобразованиях один и тот же блок или элемент ключа используется многократно. Это позволяет при выполнении условия обратимости функции относительно величины X сделать функцию необратимой относительно ключа Key.

- Поскольку операция зашифровки или расшифровки отдельного блока в процессе кодирования выполняется многократно, а значение ключа и, следовательно, функций Vi(Key) остается неизменным, то иногда становится целесообразно заранее однократно вычислить данные значения и хранить их в оперативной памяти совместно с ключом. Данная операция не изменяет ни длину ключа, ни криптостойкость алгоритма в целом. Здесь происходит лишь оптимизация скорости вычислений путем кеширования промежуточных результатов. Описанные действия встречаются практически во многих блочных криптоалгоритмах и носят название расширение ключа (англ. key scheduling)

Сеть Фейштеля



Структура сети

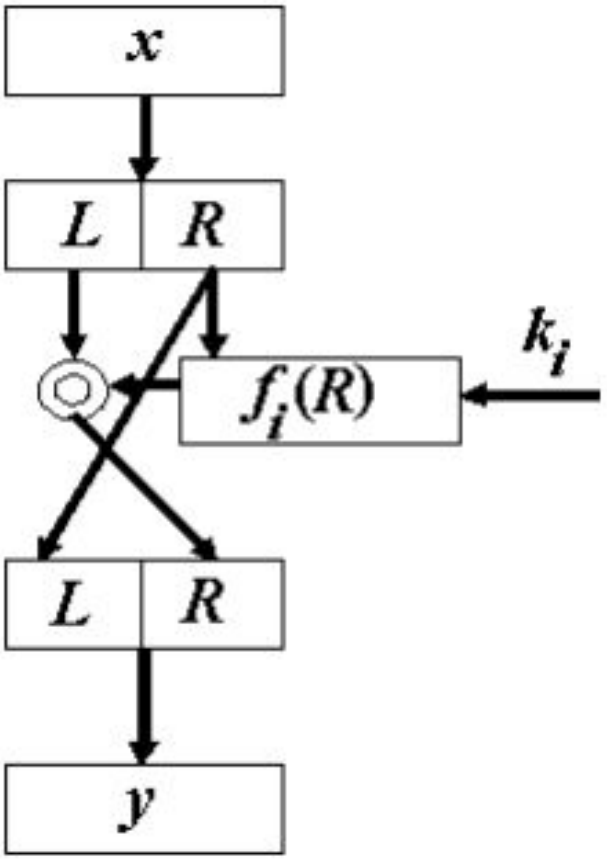
- Независимые потоки информации, порожденные из исходного блока, называются ветвями сети. В классической схеме их две.
- Величины Vi именуются параметрами сети, обычно это функции от материала ключа.
- Функция F называется образующей.
- Действие, состоящее из однократного вычисления образующей функции и последующего наложения ее результата на другую ветвь с обменом их местами, называется циклом или раундом (англ. round) сети Фейштеля. Оптимальное число раундов K – от 8 до 32.

- увеличение количества раундов значительно увеличивает криптоскойстость любого блочного шифра к криптоанализу. Возможно, эта особенность и повлияла на столь активное распространение сети Фейштеля – ведь при обнаружении, какого-либо слабого места в алгоритме, почти всегда достаточно увеличить количество раундов на 4-8, не переписывая сам алгоритм. Часто количество раундов не фиксируется разработчиками алгоритма, а лишь указываются разумные пределы (обязательно нижний, и не всегда – верхний) этого параметра.

Структура i-го раунда шифрования блока сети Фейстеля

Блок текста x длиной N разобьем на два полублока: L (левый старший) и R (правый младший):

$$x = \{L; R\}, \quad |x| = N, \quad |L| = |R| = N/2.$$



На вход раундовой функции подадим правый полублок R и раундовий подключ k_i и применим к ним операцию XOR. Для подготовки к следующему $(i+1)$-му раунду меняем местами левый и правый полублоки местами (говорят образуется ***петля Фейстеля***). Процес преобразования i-го раунда сети Фейстеля выглядит так:

$$\begin{cases} L_i = R_{i-1}; \\ R_i = L_{i-1} \oplus f(R_{i-1}; k_i). \end{cases}$$

Так как

$$L_{i-1} \oplus f(R_{i-1}; k_i) \oplus f(R_{i-1}; k_i) = L_{i-1},$$

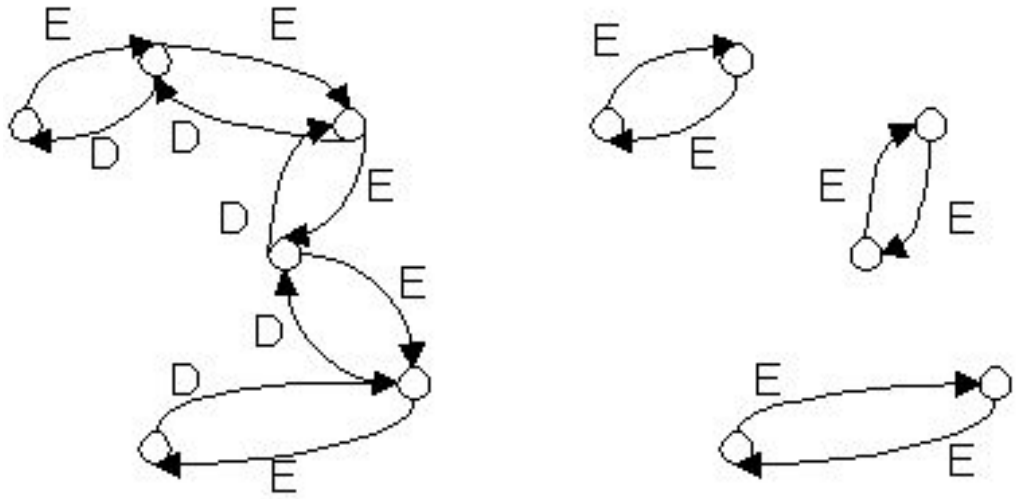
то сеть Фейстеля гарантирована обратима с учетом возможности восстановления исходных данных в каждом раунде.

- Для получения криптопреобразования, обладающего хорошими криптографическими свойствами, функция усложнения f , используемая в раунде, реализуется в виде композиции элементарных преобразований, называемых слоями функции усложнения (функцию f называют еще раундовой, цикловой). Конструктивные слои функции усложнения имеют следующие назначения: подмешивание раундовых ключей; перемешивание входных блоков; реализацию сложной нелинейной зависимости между знаками ключа, входного и выходного блоков.

Свойства схемы:

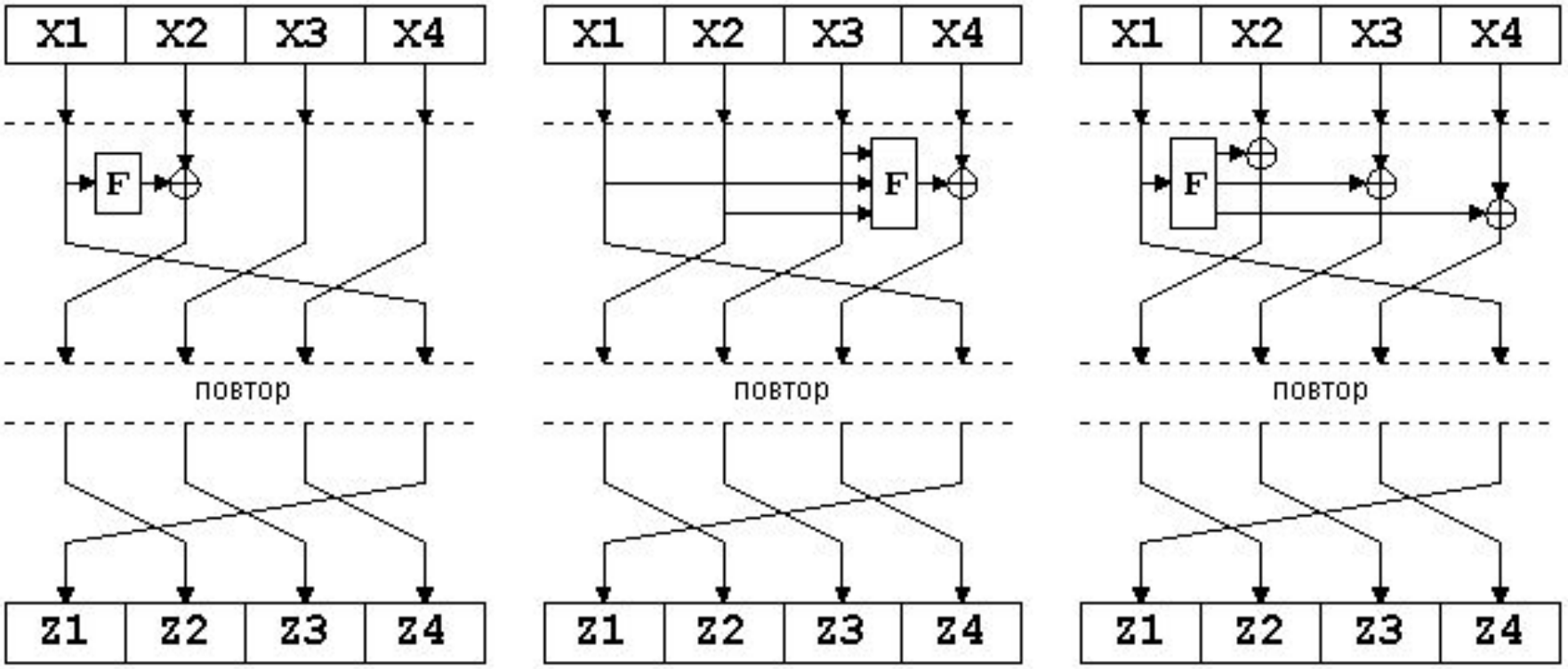
- Обратима. Даже если в качестве образующей функции F будет использовано необратимое преобразование, то и в этом случае вся цепочка будет восстановима. Это происходит вследствие того, что для обратного преобразования сети Фейштеля не нужно вычислять функцию F^{-1}.
- Симметрична. Использование операции XOR, обратимой своим же повтором, и инверсия последнего обмена ветвей делают возможным раскодирование блока той же сетью Фейштеля, но с инверсным порядком параметров Vi. Для обратимости сети Фейштеля не имеет значение является ли число раундов четным или нечетным числом. В большинстве реализаций схемы, в которых сохранены условия (операция XOR и уничтожение последнего обмена), прямое и обратное преобразования производятся одной и той же процедурой, которой в качестве параметра передается вектор величин Vi либо в исходном, либо в инверсном порядке. Сеть Фейштеля можно сделать и абсолютно симметричной, то есть выполняющей функции шифрования и дешифрования одним и тем же набором операций.

Если мы рассмотрим граф состояний криптоалгоритма, на котором точками отмечены блоки входной и выходной информации, то при каком-то фиксированном ключе для классической сети Фейштеля мы будем иметь картину, изображенную на рис.а, а во втором случае каждая пара точек получит уникальную связь, как изображено на рис. б. Модификация сети Фейштеля, обладающая подобными свойствами, характеризуется повторным использованием данных ключа в обратном порядке во второй половине цикла.



а) б)

- Модификацию сети Фейштеля для большего числа ветвей применяют гораздо чаще. Это в первую очередь связано с тем, что при больших размерах кодируемых блоков (128 и более бит) становится неудобно работать с математическими функциями по модулю 64 и выше.



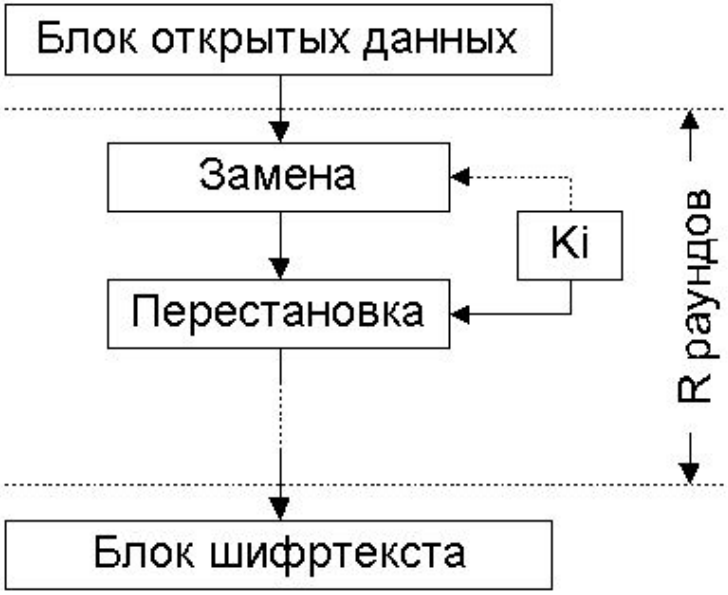
а) type 1 б) type 2 в) type 3

- Сеть Фейштеля надежно зарекомендовала себя как криптостойкая схема произведения преобразований, и ее можно найти практически в любом современном блочном шифре. Незначительные модификации касаются обычно дополнительных начальных и оконечных преобразований (англоязычный термин – whitening) над шифруемым блоком. Подобные преобразования, выполняемые обычно также либо "исключающим ИЛИ" или сложением имеют целью повысить начальную рандомизацию входного текста. Таким образом, криптостойкость блочного шифра, использующего сеть Фейштеля, определяется на 95% функцией F и правилом вычисления Vi из ключа. Эти функции и являются объектом все новых и новых исследований специалистов в области криптографии

Архитектура блочных шифров

1. Сбалансированная сеть Фейштеля – размеры ветвей равны (DES)
2. Несбалансированная сеть Фейштеля – размеры ветвей различны, функция шифрования F может зависеть не от всех битов исходного блока данных или иметь разные зависимости в разных раундах (MARS)

3. Подстановочно-перестановочная сеть обрабатывают за один раунд целиком шифруемый блок. Обработка данных сводится, в основном, к заменам в соответствии с таблицей замен, которая может зависеть от значения ключа и перестановкам, зависящим от ключа.. SP-сети являются гораздо менее распространенными (Safer+)



4. Архитектура "квадрат"(Square).

Шифруемый блок данных - в виде двумерного байтового массива. Криптографические преобразования могут выполняться над отдельными байтами массива, а также над его строками или столбцами. Недостатком алгоритмов со структурой "квадрат" является их недостаточная изученность, что не помешало алгоритму Rijndael стать новым стандартом США.

.

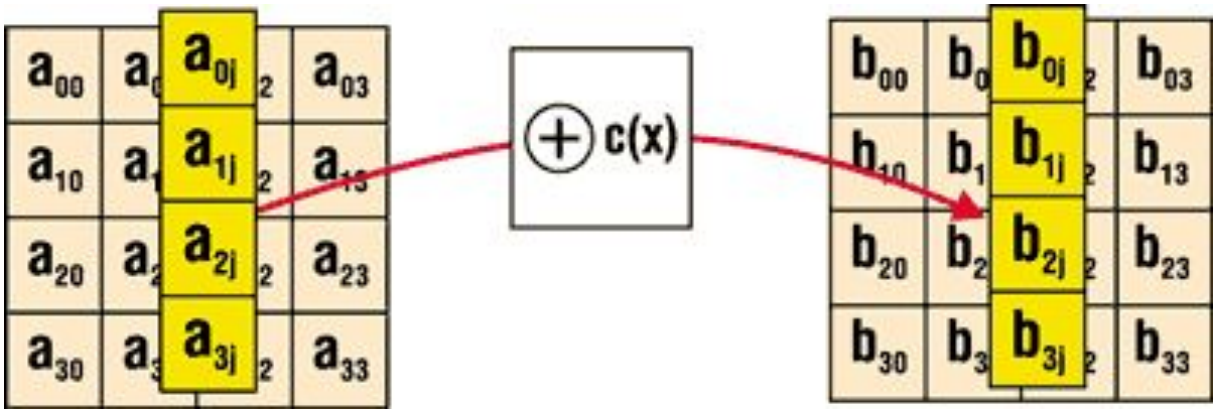
Алгоритм представляет каждый блок кодируемых данных в виде двумерного массива байт размером 4x4, 4x6 или 4x8 в зависимости от установленной длины блока.

Функция нелинейного преобразования в алгоритме Rijndael состоит из трех следующих элементарных преобразований, выполняемых последовательно:

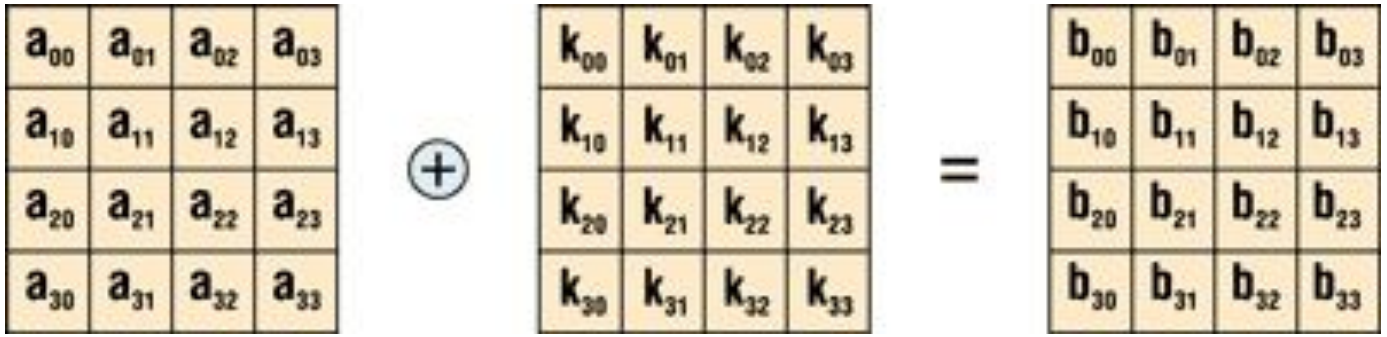
- байтовая подстановка – каждый байт преобразуемого блока заменяется новым значением, извлекаемым из общего для всех байтов матрицы вектора замены;

- побайтовый циклический сдвиг в строках матрицы: первая строка остается неизменной, вторая строка циклически сдвигается влево на один байт, третья и четвертая строка циклически сдвигаются влево соответственно на 2 и 3 байта для n = 4 или 6, и на 3 и 4 байта для n = 8;
-

- матричное умножение – полученная на предыдущем шаге матрица умножается слева на матрицу–циркулянт размера 4x4: - операцияоперация над независимыми столбцами массива когда каждый столбец по определенному правилу умножается на фиксированную матрицу c(x)



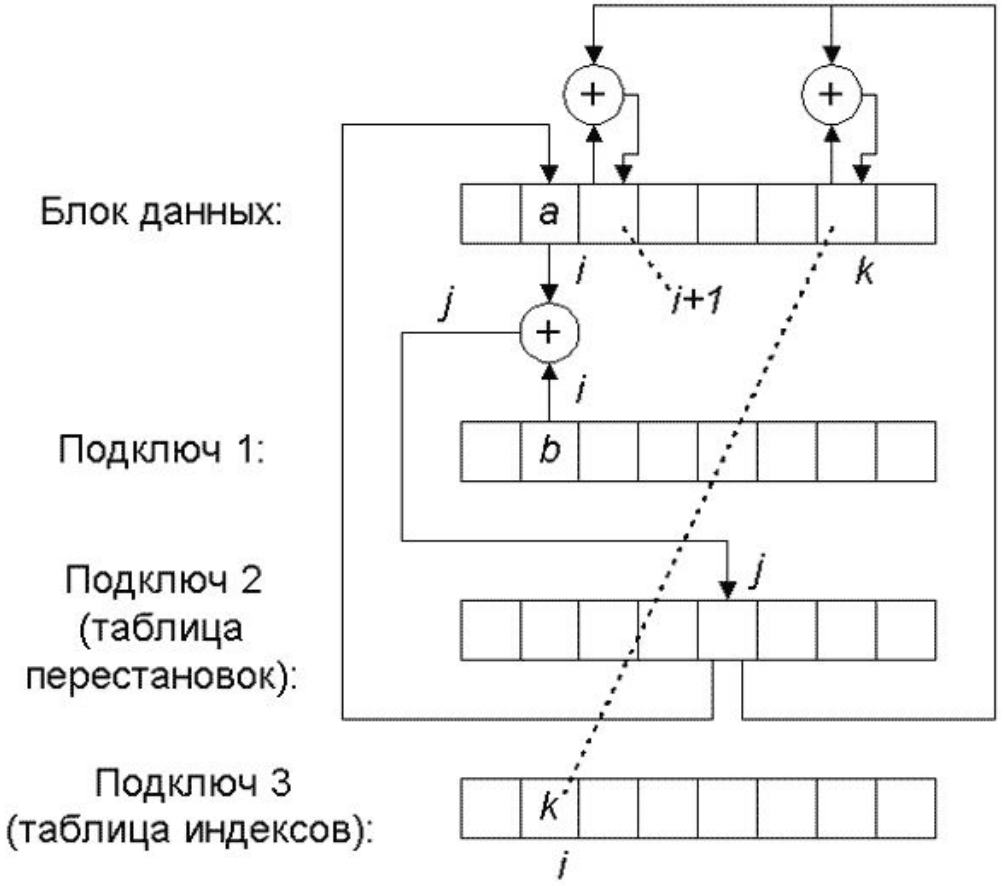
- добавление ключа. Каждый бит массива складывается по модулю 2 с соответствующим битом ключа раунда, который, в свою очередь, определенным образом вычисляется из ключа шифрования



- В каждом раунде над шифруемыми данными поочередно выполняются перечисленные преобразования.
- Расшифрование выполняется с помощью следующих обратных операций. Выполняется обращение таблицы и табличная замена на инверсной таблице (относительно применяемой при зашифровании). Обратная операция к SR - это циклический сдвиг строк вправо, а не влево. Обратная операция для MC - умножение по тем же правилам на другую матрицу d(x), удовлетворяющую условию: c(x) * d(x) = 1. Добавление ключа AK является обратным самому себе, поскольку в нем используется только операция XOR. Эти обратные операции применяются при расшифровании в последовательности, обратной той, что использовалась при зашифровании.

- Параметры:
  - размер блока 128, 192, 256 бит,;
  - размер ключа 128, 192, 256 бит
  - число раундов 10, 12, 14. Зависит от размера блока (Nb) и ключа (Nk), заданных в битах, по следующей формуле: Nr=max(Nb,Nk)/32+6;
- Rijndael стал новым стандартом шифрования данных благодаря целому ряду преимуществ перед другими алгоритмами. Прежде всего он обеспечивает высокую скорость шифрования на всех платформах: как при программной, так и при аппаратной реализации. Его отличают несравнимо лучшие возможности распараллеливания вычислений по сравнению с другими алгоритмами, представленными на конкурс. Кроме того, требования к ресурсам для его работы минимальны, что важно при его использовании в устройствах, обладающих ограниченными вычислительными возможностями

5. Алгоритмы с нестандартной структурой В качестве примера алгоритма с нестандартной структурой можно привести уникальный по своей структуре алгоритм FROG, в каждом раунде которого по достаточно сложным правилам выполняется модификация двух байт шифруемых данных



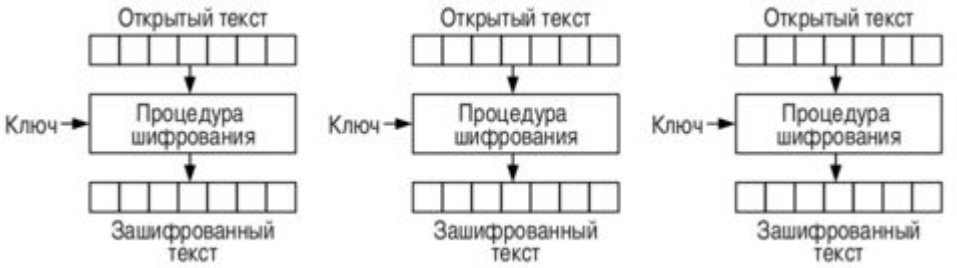
Режимы шифрования

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

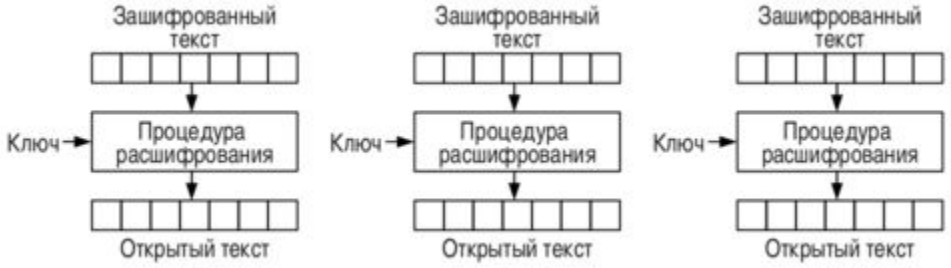
ЭЛЕКТРОННАЯ КОДОВАЯ КНИГА

- Каждый блок шифруют независимо от других с использованием одного ключа шифрования.

Шифрование в режиме ECB



Расшифрование в режиме ECB



режим применяется для шифрования небольших объемов информации, размером не более одного блока или для шифрования ключей. Это связано с тем, что одинаковые блоки открытого текста преобразуются в одинаковые блоки шифротекста, что может дать взломщику (криптоаналитику) определенную информацию о содержании сообщения. Основным достоинством этого режима является простота реализации.

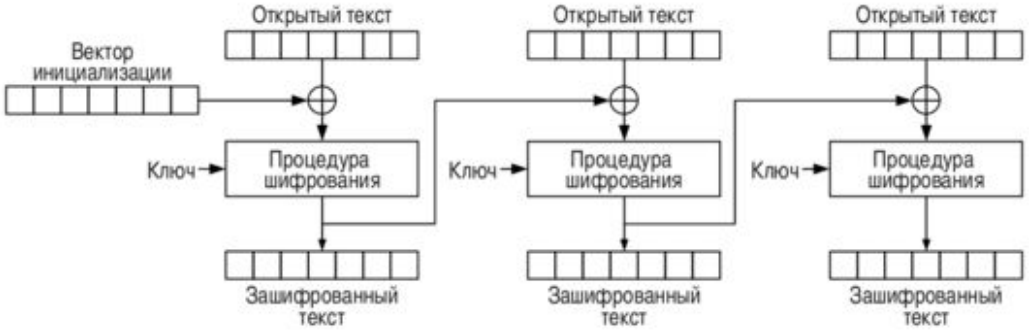
СЦЕПЛЕНИЕ БЛОКОВ ШИФРОТЕКСТА

- Исходный текст разбивается на блоки, а затем обрабатывается по следующей схеме:
- Первый блок складывается побитно по модулю 2 (XOR) с неким значением IV - начальным вектором (Init Vector), который выбирается независимо перед началом шифрования. Полученное значение шифруется. Полученный в результате блок шифротекста отправляется получателю и одновременно служит начальным вектором IV для следующего блока открытого текста. Расшифрование осуществляется в обратном порядке.

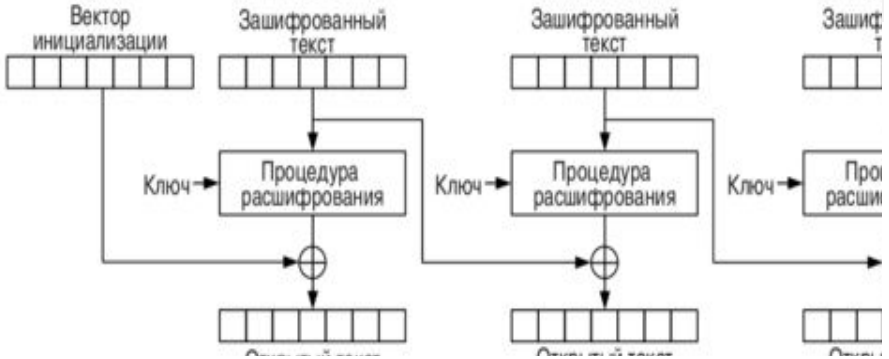
$C_0 = IV \qquad C_i = E_k(P_i \oplus C_{i-1})$

$P_i = C_{i-1} \oplus D_k(C_i)$

Шифрование в режиме CBC



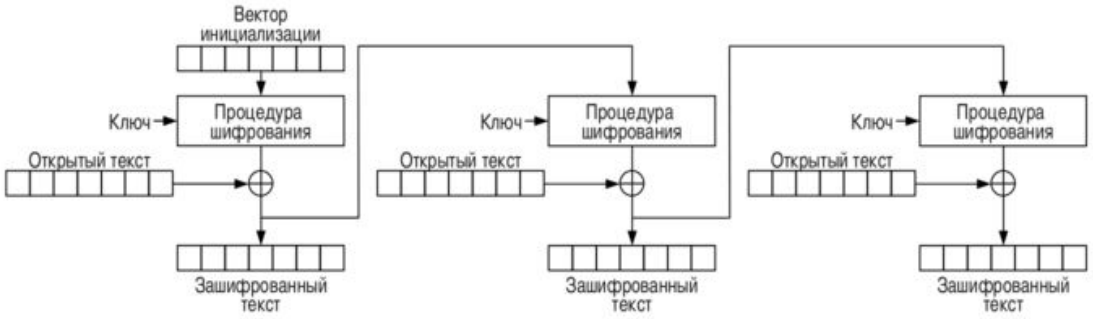
Расшифрование в режиме CBC



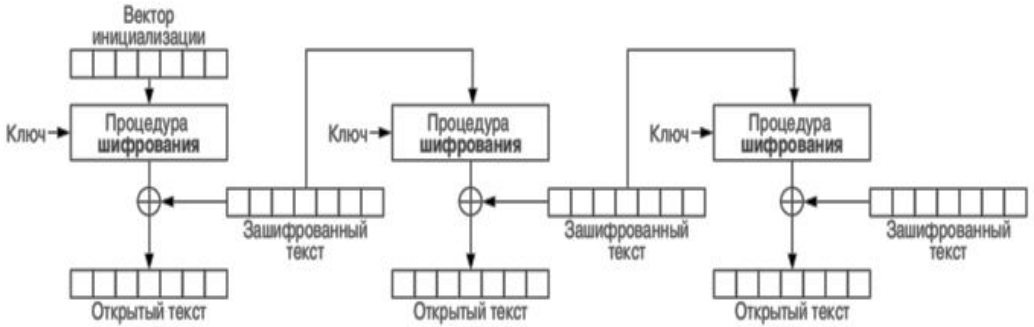
Типичные приложения - общая блокоориентированная передача, аутентификация.

ОБРАТНАЯ СВЯЗЬ ПО ШИФРОТЕКСТУ

- *Шифрование в режиме CFB*



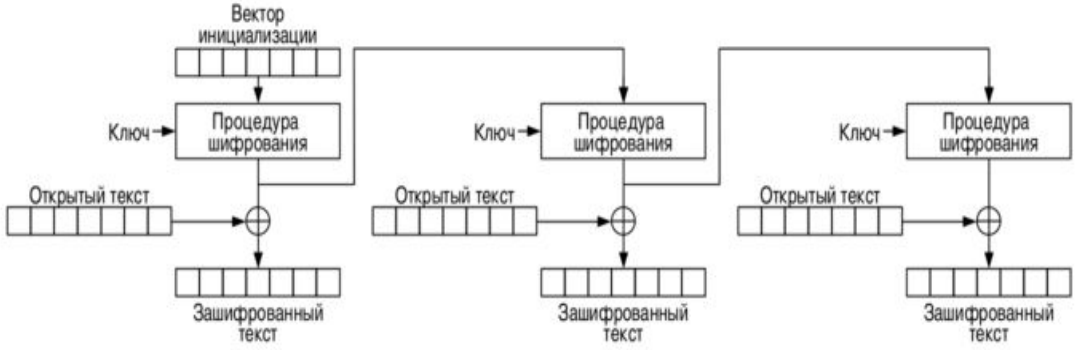
Расшифрование в режиме CFB

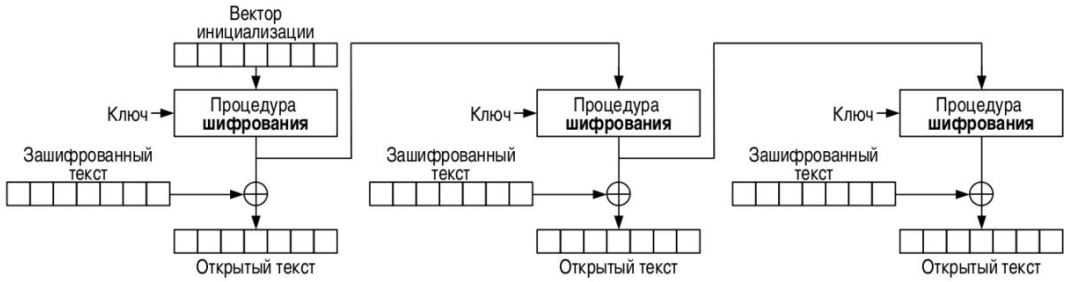


$$C_0 = IV$$
$$C_i = E_k\left(C_{i-1}\right) \oplus P_i$$
$$P_i = E_k\left(C_{i-1}\right) \oplus C_i$$

Вначале IV заполняется неким значением, которое называется синхропосылкой, не является секретным и передается перед сеансом связи получателю.
Значение IV шифруется и складыватся (XOR) с открытым текстом и получается блок шифротекста.
Значением IV становится значение ш.т. Расшифрование аналогично.
Особенностью данного режима является распространение ошибки на весь последующий текст. Применяется как правило для шифрования потоков информации типа оцифрованной речи, видео.

ОБРАТНАЯ СВЯЗЬ ПО ВЫХОДУ





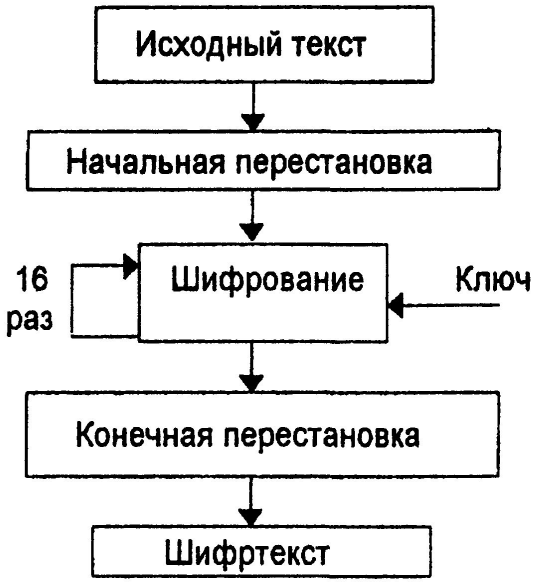
- $O_0 = IV \quad O_i = E_k(O_i - 1)$

$$C_i = P_i \oplus O_i$$

$$P_i = C_i \oplus O_i$$

Принцип работы схож с принципом работы режима CFB, но сдвиговый регистр IV заполняется не битами шифротекста, а шифрованными битами ключа Расшифрование осуществляется аналогично. Главное свойство шифра - единичные ошибки не распространяются, т.к заполнение сдвигового регистра осуществляется не зависимо от шифротекста. Область применения: потоки видео, аудио или данных, для которых необходимо обеспечить оперативную доставку. Широко используется у военных наряду с поточными шифрами.

Алгоритм DES



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

При этом бит 58 блока T становится битом 1, бит 50 — битом 2 и т. д. Полученный после перестановки блок $IP(T)$ разделяется на две половины: L_0, состоящую из 32 старших бит, и R_0, состоящую из 32 младших бит.

Затем выполняется итеративный процесс шифрования, состоящий из 16 циклов преобразования Фейстеля. Пусть $T_{i-1} = L_{i-1}R_{i-1}$ — результат $(i-1)$-й итерации.

Тогда результат i-й итерации $T_i = L_iR_i$ определяется формулами

$$\begin{cases} L_i = R_{i-1}, \\ R_i = L_{i-1} \oplus f(R_{i-1}, k_i), \quad i = \overline{1,16}. \end{cases} \tag{2}$$

Функция f называется *функцией шифрования*. Ее аргументами являются 32-битовый вектор R_{i-1} и 48-битовый ключ k_i, который является результатом преобразования 56-битового ключа шифра k. Результатом последней итерации является блок $T_{16} = R_{16}L_{16}$. По окончании шифрования осуществляется восстановление позиций битов применением к T_{16} обратной перестановки IP^{-1}.

При расшифровании данных все действия выполняются в обратном порядке, при этом вместо соотношений (2) следует использовать соотношения

$$\begin{cases} R_{i-1} = L_i, \\ L_{i-1} = R_i \oplus f(L_i, k_i), \quad i = \overline{16,1}, \end{cases}$$

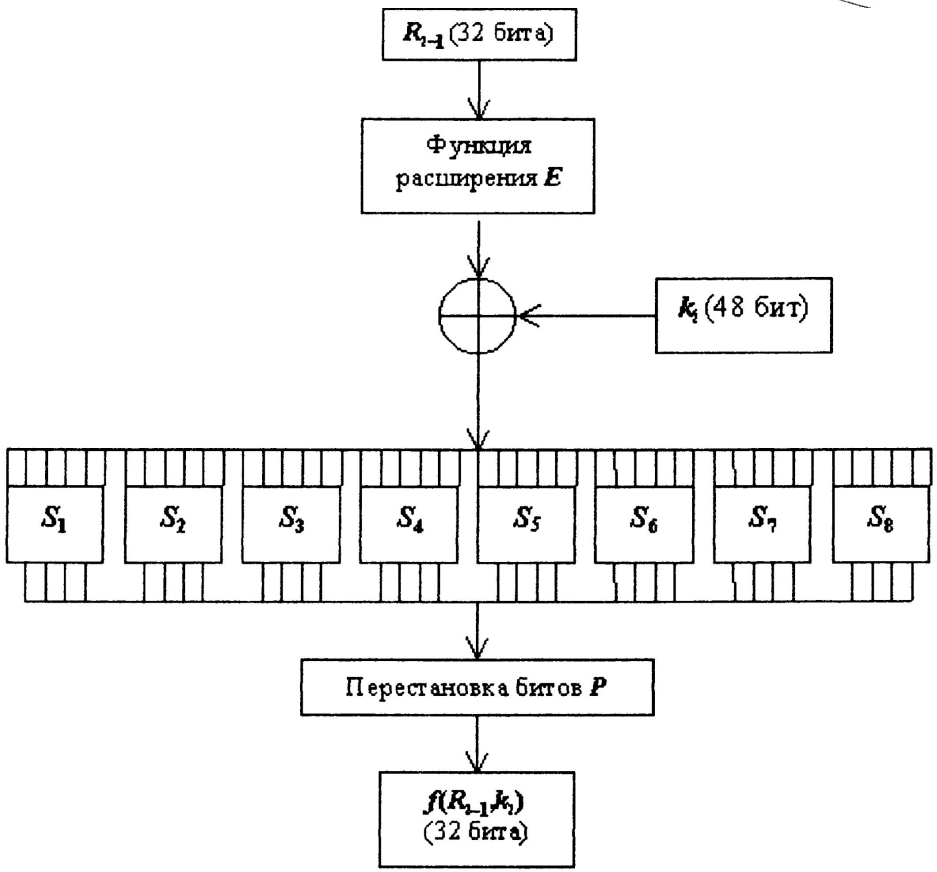
Для вычисления значения функции f используются: *функция расширения* E; преобразование S, составленное из восьми преобразований S-*блоков* $S_1, S_2, ..., S_8$; перестановка P. Аргументами функции f являются вектор R_{i-1} (32 бита) и вектор k_i (48 бит). Функция E "расширяет" 32-битовый вектор R_{i-1} до 48-битового вектора $E(R_{i-1})$ путем дублирования некоторых битов вектора R_{i-1}, при этом

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

Полученный результат складывается побитно по модулю 2 с текущим значением ключа k_i и затем представляется в виде восьми последовательных 6-битовых блоков $B_1,\ldots,B_8$:

$$E(R_{i-1}) \oplus k_i = B_1 \ldots B_8 .$$

Далее каждый из блоков B_j трансформируется в 4-битовый блок B'_j с помощью подходящей таблицы S-блока S_j,



Преобразование блока B_j в B_j' осуществляется следующим образом. Пусть, например, B_2 равен 111010. Первый и последний разряды B_2 являются двоичной записью числа a, $0 \le a \le 3$. Аналогично средние 4 разряда представляют число b, $0 \le b \le 15$. В нашем примере $a = 2$, $b = 13$.

Строки и столбцы таблицы S_2 пронумерованы числами a и b. Таким образом, пара (a, b) однозначно определяет число, находящееся на пересечении строки с номером a и столбца с номером b. В данном случае это число равно 10

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	4	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	

Значение $f(R_{i-1},k_i)$ теперь получается применением перестановки битов P, заданной таблицей к результирующему 32-битовому блоку $B_1' B_2' \ldots B_8'$.

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

На каждой итерации используется текущее значение ключа k_i (48 бит), получаемое из исходного ключа k следующим образом.

Сначала пользователи выбирают сам ключ k, содержащий 56 случайных значащих битов. Восемь битов, находящихся в позициях 8,16,...,64, добавляются в ключ таким образом, чтобы каждый байт содержал нечетное число единиц.

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

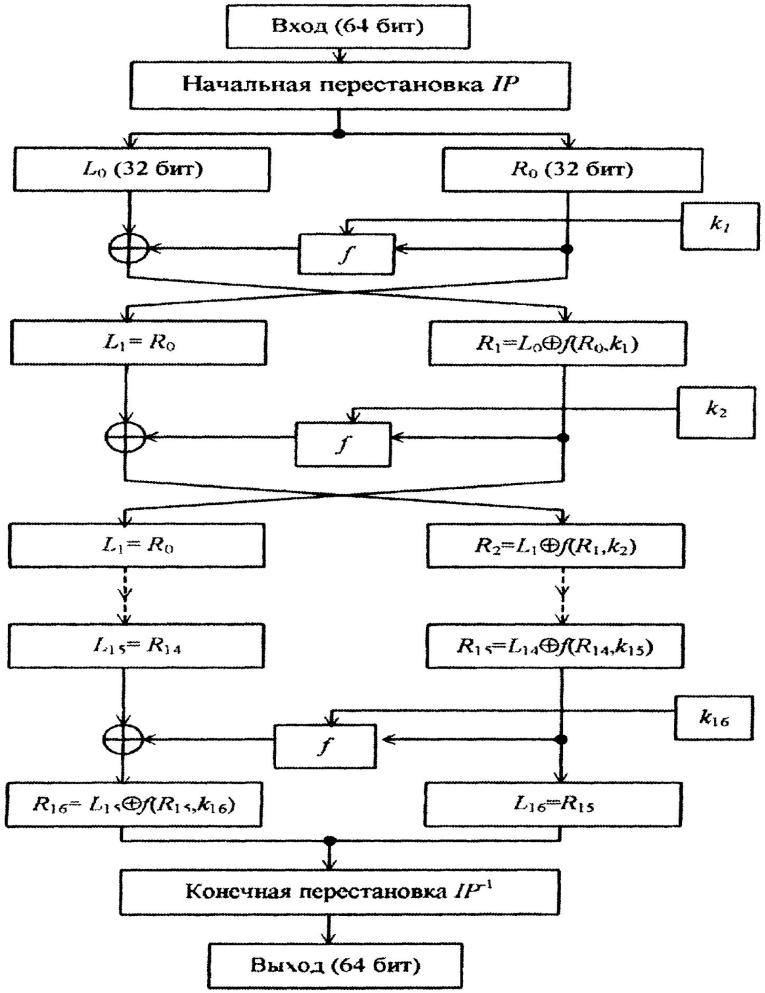
Эта перестановка определяется двумя блоками C_0 и D_0 по 28 бит в каждом (они занимают соответственно верхнюю и нижнюю половины таблицы). Так, первые три бита в C_0 есть соответственно 57, 49, 41 биты ключа. Затем индуктивно определяются блоки C_i и D_i, $i = \overline{1,16}$.

Если уже определены C_{i-1} и D_{i-1}, то C_i и D_i получаются из них одним или двумя левыми циклическими сдвигами

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число сдвигов	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Теперь определим ключи k_i, $1 \le i \le 16$. Ключ k_i состоит из 48 битов, выбираемых из битов блока C_iD_i согласно таблице 15. Первыми тремя битами в k_i являются биты 14, 17, 11 из блока C_iD_i. Отметим, что 8 из 56 бит (с номерами 9, 18, 22, 25, 35, 38, 43, 54) из C_iD_i отсутствуют в k_i.

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 ANSI X9.17

- Этот алгоритм является национальным стандартом США для генерации двоичной псевдослучайной последовательности. Используется в приложениях, обеспечивающих безопасность финансовых платежей, и PGP. В этом генераторе в качестве односторонней функции используется алгоритм шифрования TripleDES (3DES, «тройной DES»). Этот алгоритм использует два 64-битных ключа и следующим образом:

$$F_K(M) = E_{K_1}\left(D_{K_2}\left(E_{K_1}(M)\right)\right),$$

где $K = K_1 \,||\, K_2$ – составной 128-битный ключ, $E_{K_1}(M)$ – шифрование сообщения M алгоритмом DES с ключом K_1, $D_{K_2}(M)$ – дешифрование сообщения M алгоритмом DES с ключом $K_2 \neq K_1$.

- Входные данные: некоторое случайное (и секретное) 64-битное начальное значение s_0, 128-битный составной ключ K и m – количество генерируемых 64-битных двоичных слов.
- Выходные данные: последовательность m 64-битных двоичных слов $x_1, x_2, \ldots, x_m$

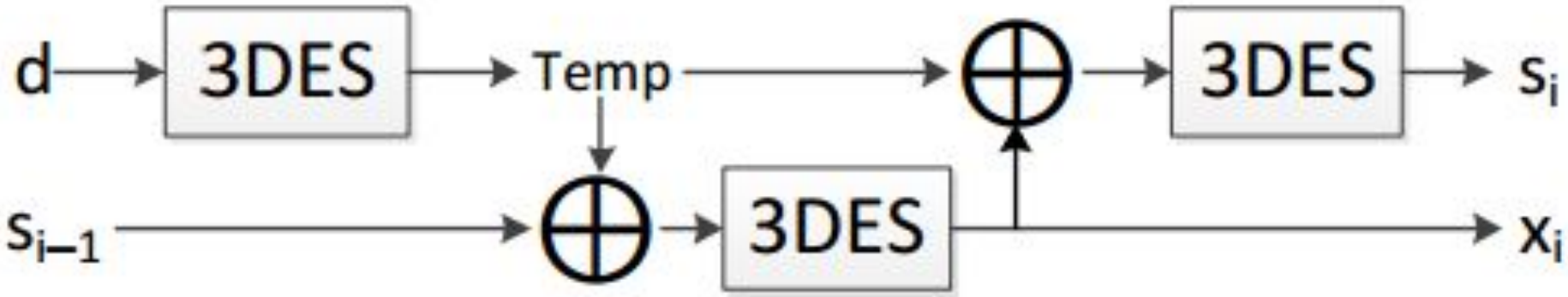


Рисунок 1 – Алгоритм ANSI X9.17

Шаги алгоритма:

1. Зафиксировать 64-битное представление d даты и времени в момент обращения к программе генерации и вычислить вспомогательное 64-битное двоичное слово $Temp = F_K(d)$.
2. Для $i = \overline{1,m}$:
   1) вычислить значение i-го выходного слова $x_i = F_K(Temp \oplus s_{i-1})$;
   2) вычислить новое значение параметра s_i: $s_i = F_K(x_i \oplus Temp)$.
3. В результате предыдущего шага формируется выходная псевдослучайная последовательность из m слов $x_1, x_2, \dots, x_m$ либо двоичная псевдослучайная последовательность из $64 \cdot m$ бит: $X = x_1 \,||\, x_2 \,||\, \dots \,||\, x_m$.

Примечание: алгоритм шифрования *DES* можно не реализовывать самостоятельно, а использовать готовую реализацию.