Стандарт шифрования данных DES (Data Encryption Standard)

Стандарт шифрования данных DES, который ANSI называет Алгоритмом шифрования данных DEA (Data Encryption Algorithm), а ISO - DEA-1, стал мировым стандартом.

15. 05.1973 года были опубликованы требования к криптографическому алгоритму, который мог бы быть принят в качестве стандарта.: - Алгоритм должен обеспечивать высокий уровень безопасности.

-Алгоритм должен быть полностью определен и легко понятен.

- Безопасность алгоритма должна основываться на ключе и не должна зависеть от сохранения в тайне самого алгоритма.

- Алгоритм должен быть доступен всем пользователям.

- Алгоритм должен позволять адаптацию к различным применениям.

- Алгоритм должен позволять экономичную реализацию в виде электронных приборов.

-Алгоритм должен быть эффективным в использовании.

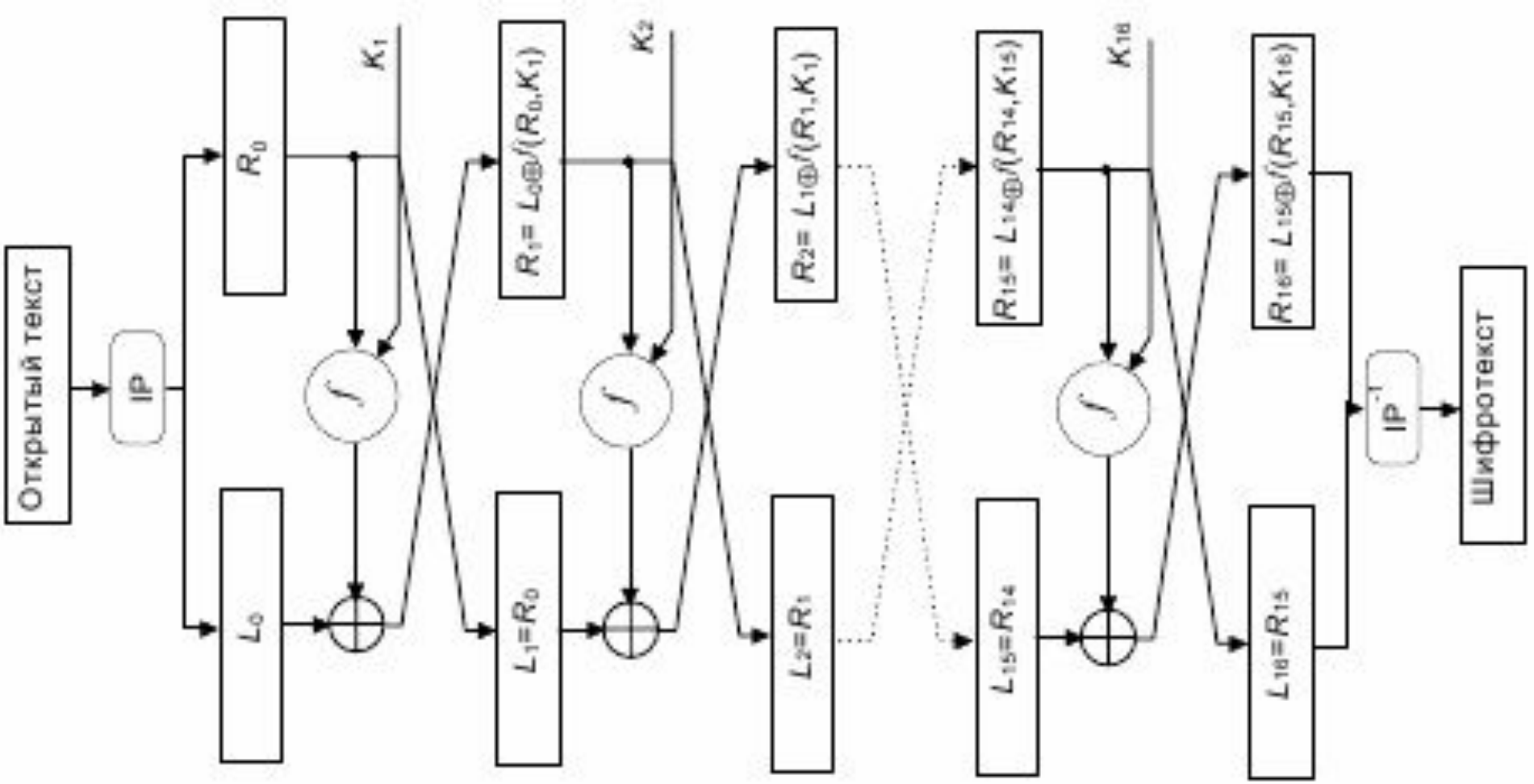
- Алгоритм должен предоставлять возможности проверки.

- Алгоритм должен быть разрешен для экспорта.

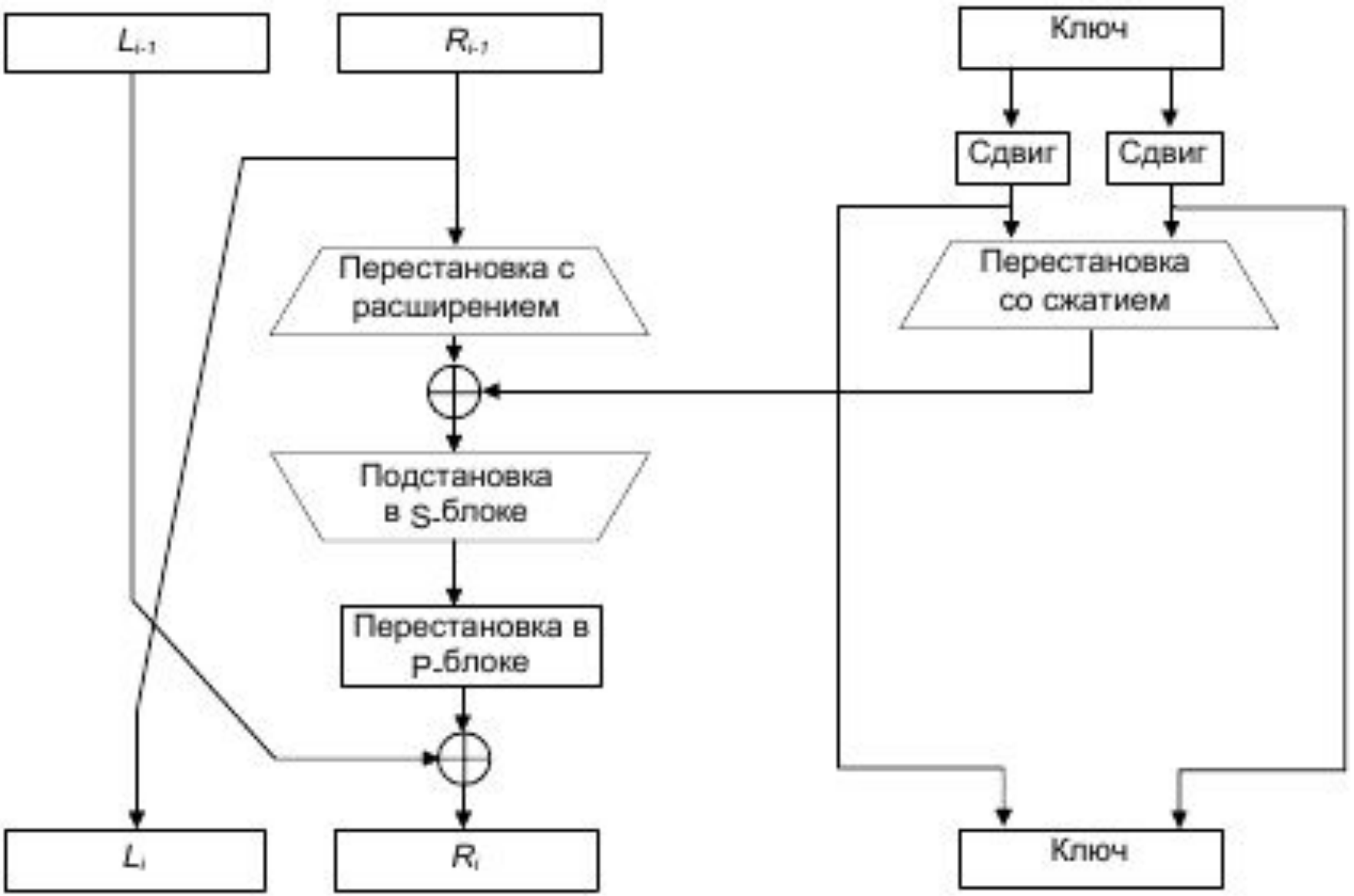
В IBM существовала целая команда криптографов, работавшая в Кингстоне (Kingston) и Йорктаун Хайтс (Yorktown Heights), в которую входили Рой Адлер (Roy Adler), Дон Копперсмит (Don Coppersmith), Хорст Фейстель (Horst Feistel), Эдна Кроссман (Edna Crossman), Алан Конхейм (Alan Konheim), Карл Майер (Carl Meyer), Билл Ноц (Bill Notz), Линн Смит (Lynn Smith), Уолт Тачмен (Walt Tuchman) и Брайант Такерман (Bryant Tuckerman, первый. алгоритм IBM – Люцифер. Несмотря на критику Стандарт шифрования данных DES 23 ноября 1976 года был принят в качестве федерального стандарта и разрешен к использованию на всех несекретных правительственных коммуникациях. Официальное описание стандарта, FIPS PUB 46, "Data Encryption Standard", было опубликовано 15/01/1977 года и вступило в действие шестью месяцами позже

DES представляет собой блочный шифр, он шифрует данные 64-битовыми блоками. На входе алгоритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый блок шифротекста. DES является симметричным алгоритмом: для шифрования и дешифрирования используются одинаковые алгоритм и ключ (за исключением небольших различий в использовании ключа). Длина ключа равна 56 битам. (Ключ обычно представляется 64-битовым числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа.) Ключ, который может быть любым 56-битовым числом, можно изменить в любой момент времени. Ряд чисел считаются слабыми ключами, но их можно легко избежать. Безопасность полностью определяется ключом. На простейшем уровне алгоритм - комбинация двух основных методов шифрования: смещения и диффузии. Фундаментальным строительным блоком DES является применение к тексту единичной комбинации этих методов (подстановка, а за ней - перестановка), зависящей от ключа. Такой блок называется этапом.

DES состоит из 16 этапов, одинаковая комбинация методов применяется к открытому тексту 16 раз.



DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняется 16 этапов одинаковых действий, называемых функцией f, в которых данные объединяются с ключом. После шестнадцатого этапа правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной). На каждом этапе биты ключа сдвигаются, и затем из 56 битов ключа выбираются 48 битов. Правая половина данных увеличивается до 48 битов с помощью перестановки с расширением, объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через 8 S-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией f. Затем результат функции f объединяется с левой половиной с помощью другого XOR. В итоге этих действий появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 этапов DES.



Один из этапов DES

Если B i - это результат i-ой итерации, L i и R i - левая и правая половины B i, K i - 48-битовый ключ для этапа i, а f - это функция, выполняющие все подстановки, перестановки и XOR с ключом, то этап можно представить как:

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

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

Начальная перестановка выполняется еще до этапа 1, при этом входной блок переставляется, как показано в таблице. Эту и все другие таблицы этой главы надо читать слева направо и сверху вниз. Например, начальная перестановка перемещает бит 58 в битовую позицию 1, бит 50 - в битовую позицию 2, бит 42 - в битовую позицию 3, и так далее.

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

Начальная перестановка и соответствующая заключительная перестановка не влияют на безопасность DES

Преобразования ключа Сначала 64-битовый ключ DES уменьшается до 56-битового ключа отбрасыванием каждого восьмого бита. Эти биты используются только для контроля четности, позволяя проверять правильность ключа. После извлечения 56-битового ключа для каждого из 16 этапов DES генерируется новый 48-битовый подключ. Эти подключи, K i, определяются следующим образом.

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

Во первых, 56-битовый ключ делится на две 28-битовых половинки. Затем, половинки циклически сдвигаются налево на один или два бита в зависимости от этапа. Этот сдвиг показан в таблице

Число битов сдвига ключа в зависимости от этапа

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

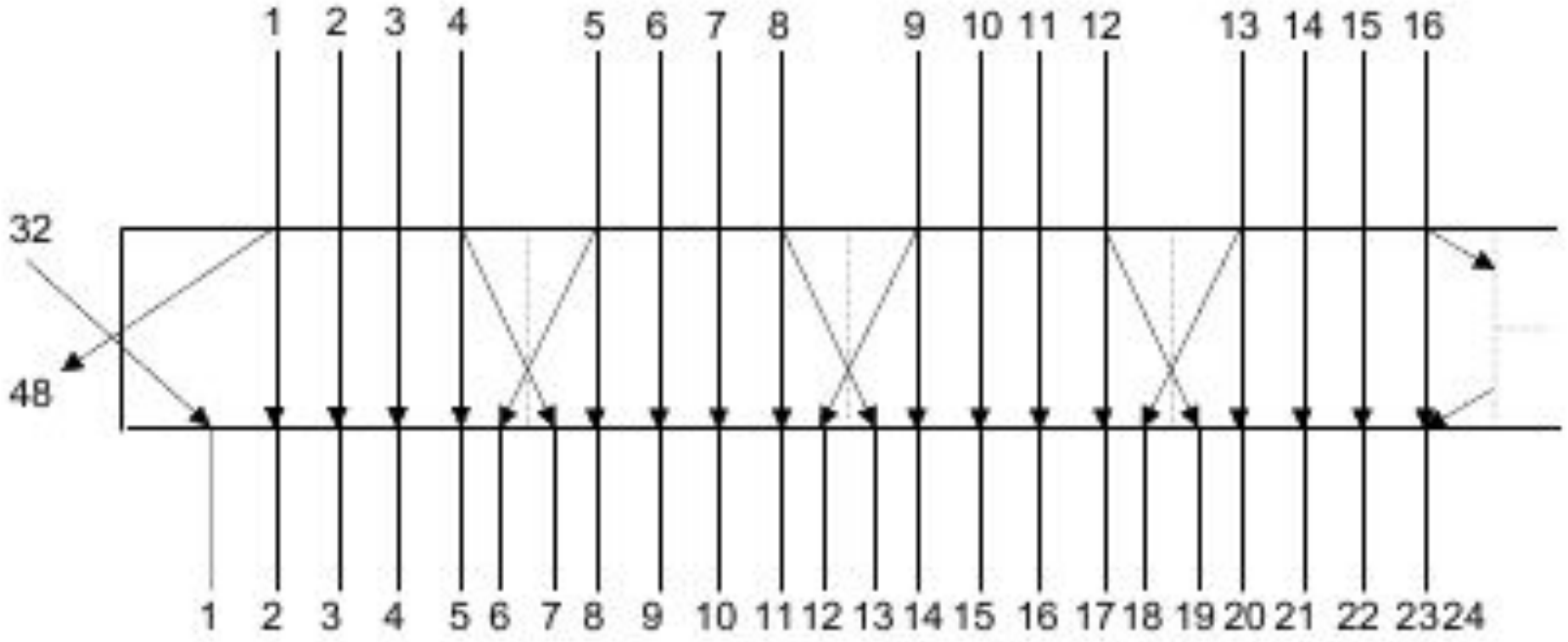
После сдвига выбирается 48 из 56 битов. Так как при этом не только выбирается подмножество битов, но и изменяется их порядок, эта операция называется перестановка со сжатием. Ее результатом является набор из 48 битов. Перестановка со сжатием (также называемая переставленным выбором) определена в таблице. Из-за сдвига для каждого подключа используется отличное подмножество битов ключа. Каждый бит используется приблизительно в 14 из 16 подключей, хотя не все биты используются в точности одинаковое число раз.

Перестановка со сжатием

14,	17,	11,	2,4,	1,	5,	3,	28,	15,	6,	21,	10,
23,	19,	11,	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

Перестановка с расширением Эта операция расширяет правую половину данных, R i, от 32 до 48 битов. Так как при этом не просто повторяются определенные биты, но и изменяется их порядок, эта операция называется перестановкой с расширением.

У нее две задачи: привести размер правой половины в соответствие с ключом для операции XOR и получить более длинный результат, который можно будет сжать в ходе операции подстановки. Однако главный криптографический смысл совсем в другом. За счет влияния одного бита на две подстановки быстрее возрастает зависимость битов результата от битов исходных данных. Это называется **лавинным эффектом**. DES спроектирован так, чтобы как можно быстрее добиться зависимости каждого бита шифротекста от каждого бита открытого текста и каждого бита ключа. Перестановка с расширением иногда называется E-блоком (от expansion). Для каждого 4-битового входного блока первый и четвертый бит представляют собой два бита выходного блока, а второй и третий биты - один бит выходного блока. В таблице 7-й показано, какие позиции результата соответствуют каким позициям исходных данных. Например, бит входного блока в позиции 3 переместится в позицию 4 выходного блока, а бит входного блока в позиции 21 - в позиции 30 и 32 выходного блока. Хотя выходной блок больше входного, каждый входной блок генерирует уникальный выходной блок.



Перестановка с расширением

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

Подстановка с помощью S-блоков После объединения сжатого блока с расширенным блоком с помощью XOR над 48-битовым результатом выполняется операция подстановки. Подстановки производятся в восьми блоках подстановки, или S-блоках (от substitution). У каждого S-блока 6-битовый вход и 4-битовый выход, всего используется восемь различных S-блоков. (Для восьми S-блоков DES потребуется 256 байтов памяти.) 48 битов делятся на восемь 6-битовых подблоков. Каждый отдельный подблок обрабатывается отдельным S-блоком: Первый подблок - S-блоком 1, второй - S-блоком 2, и так далее. См. 8-й.

Каждый S-блок представляет собой таблицу из 2 строк и 16 столбцов. Каждый элемент в блоке является 4- битовым числом. По 6 входным битам S-блока определяется, под какими номерами столбцов и строк искать выходное значение. Все восемь S-блоков показаны в 6-й.

S-блоки

S-блок 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 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 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

Конечно же, намного легче реализовать S-блоки программно в виде массивов с 64 элементами. Для этого потребуется переупорядочить элементы, что не является трудной задачей. (Изменить индексы, не изменяя порядок элементов, недостаточно. S-блоки спроектированы очень тщательно.) Однако такой способ описания S- блоков помогает понять, как они работают. Каждый S-блок можно рассматривать как функцию подстановки 4- битового элемента: b 2 по b 5являются входом, а некоторое 4-битовое число - результатом. Биты b 1 и b 6 определяются соседними блоками, они определяют одну из четырех функций подстановки, возможных в данном S-блоке. Подстановка с помощью S-блоков является ключевым этапом DES. Другие действия алгоритма линейны и легко поддаются анализу

. S-блоки нелинейны, и именно они в большей степени, чем все остальное, обеспечивают безопасность DES. В результате этого этапа подстановки получаются восемь 4-битовых блоков, которые вновь объединяются в единый 32-битовый блок. Этот блок поступает на вход следующего этапа - перестановки с помощью P-блоков. Перестановка с помощью P-блоков 32-битовый выход подстановки с помощью S-блоков, перетасовываются в соответствии с P-блоком. Эта перестановка перемещает каждый входной бит в другую позицию, ни один бит не используется дважды, и ни один бит не игнорируется. Этот процесс называется прямой перестановкой или просто перестановкой. Позиции, в которые перемещаются биты, показаны в 5-й. Например, бит 21 перемещается в позицию 4, а бит 4 - в позицию 31. результат перестановки с помощью P-блока объединяется посредством XOR с левой половиной первоначального 64-битового блока. Затем левая и правая половины меняются местами

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

Заключительная перестановка является обратной по отношению к начальной перестановке. Левая и правая половины не меняются местами после последнего этапа DES, вместо этого объединенный блок R 16L 16 используется как вход заключительной перестановки. Это сделано для того, чтобы алгоритм можно было использовать как для шифрования, так и для дешифрирования.

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

Дешифрирование DES Различные компоненты DES были подобраны так, чтобы выполнялось очень полезное свойство: для шифрования и дешифрирования используется один и тот же алгоритм. DES позволяет использовать для шифрования или дешифрирования блока одну и ту же функцию. Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке. То есть, если на этапах шифрования использовались ключи K 1, K 2, K 3, ..., K 16, то ключами

дешифрирования будут K 16, K 15, K 14, ..., K 1. Алгоритм, который создает ключ для каждого этапа, также цикличен. Ключ сдвигается направо, а число позиций сдвига равно 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

FIPS PUB 81 определяет **четыре режима работы**: ECB, CBC, OFB и CFB. Банковские стандарты ANSI определяют для шифрования ECB и CBC, а для проверки подлинности - CBC и n-битовый CFB **Режим электронной шифровальной книги** (electronic codebook, **ECB**) - это наиболее очевидный способ использовать блочный шифр: блок открытого текста заменяется блоком шифротекста. Так как 1блок открытого текста заменяется одним и тем же блоком шифротекста, то теоретически возможно создать шифровальную книгу блоков открытого текста и соответствующих шифротекстов . Однако, если размер блока - 64 бита, то кодовая книга будет состоять из 2^{64} записей - слишком много для предварительного вычисления и хранения. Для каждого ключа понадобится отдельная шифровальная книга . Особенно уязвимы начало и окончание сообщений, где находится информация об отправителе, получателе дате и т.д - проблема стандартные заголовки и окончания

Положительной стороной является возможность шифровать несколько сообщений одним ключом без снижения безопасности. По сути, каждый блок можно рассматривать как отдельное сообщение, шифрованное тем же самым ключом. При дешифрировании битовые ошибки в шифротексте приводят к неправильному дешифрированию соответствующего блока открытого текста, но не влияет на остальной открытый текст .

В **режиме сцепления блоков шифра** (cipher block chaining**, CBC**) перед шифрованием над открытым текстом и предыдущим блоком шифротекста выполняется операция XOR. Сцепление добавляет к блочному шифру механизм обратной связи: результаты шифрования предыдущих блоков влияют на шифрование текущего блока. Другими словами, каждый блок используется для изменения шифрования следующего блока. Каждый блок шифротекста зависит не только от шифруемого блока открытого текста, но и от всех предыдущих блоков открытого текста. Режим CBC характеризуется прямой обратной связью шифротекста при шифровании и инверсной обратной связью шифротекста при дешифрировании.

При этом приложения должны уметь бороться с ошибками . Единственная битовая ошибка в блоке открытого текста повлияет на данный блок шифротекста и все последующие блоки шифротекста. В режиме CBC ошибка одного бита шифротекста влияет на один блок и один бит восстановленного открытого текста. Блок, соответствующий содержащему ошибку блоку шифротекста, искажается полностью.

В следующем блоке искажается единственный бит, находящийся в той же позиции, что и ошибочный бит. Это свойство превращения малой ошибки шифротекста в большую ошибку открытого текста называется распространением ошибки. Это является главным недостатком. Эта ошибка не влияет на блоки, расположенные через один от испорченного и далее, поэтому режим CBC является самовосстанавливающимся. Ошибка влияет на два блока, но система продолжает работать правильно для всех последующих блоков . CBC представляет собой пример блочного шифра, используемого в самосинхронизирующейся манере, но только на блоковом уровне.

Потоковые шифры преобразуют открытый текст в шифротекст по одному биту за операцию. Генератор потока ключей (иногда-генератор с бегущим ключом) выдает поток битов: k 1, k 2, k 3, ..., k i. Этот поток ключей (бегущий ключ) и поток битов открытого текста, p 1, p 2, p 3, ..., p i, подвергаются операции "исключающее или", и в результате получается поток битов шифротекста. c i =p i ⊕ k i При дешифрировании операция XOR выполняется над битами шифротекста и тем же самым потоком ключей для восстановления битов открытого текста. p i = c i ⊕ k i Так как p i ⊕ k i ⊕ k i= p i это работает правильно. Безопасность системы полностью зависит от свойств генератора потока ключей . Если генератор потока ключей выдает бесконечную строку нулей, шифротекст будет совпадать с открытым текстом, и все операция будет бессмысленна. Если генератор потока ключей выплевывает повторяющийся 16-битовый шаблон, алгоритм будет являться простым XOR с пренебрежимо малой безопасностью. Если генератор потока ключей выплевывает бесконечный поток случайных (а не псевдослучайных) битов, получается одноразовый блокнот и идеальная безопасность.

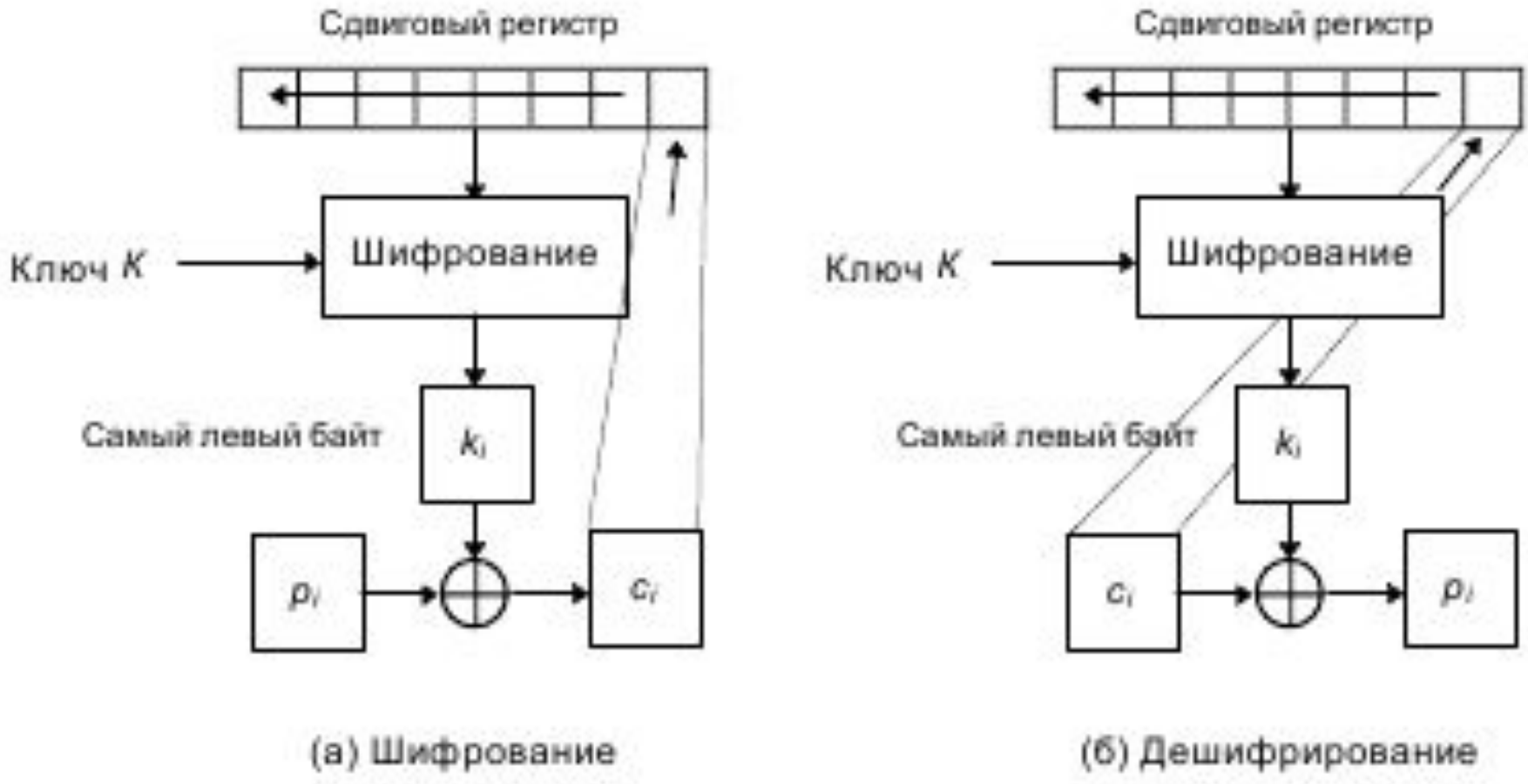
На деле безопасность потокового шифра находится где-то между простым XOR и одноразовым блокнотом.
В самосинхронизирующихся потоковых шифрах каждый бит потока ключей является функцией фиксированного числа предыдущих битов шифротекста. Военные называют этот шифр автоключом шифротекста (ciphertext auto key, CTAK). Самосинхронизирующиеся потоковые шифры также чувствительны к вскрытию повторной передачей.
В **синхронном потоковом шифре** поток ключей генерируется независимо от потока сообщения. Военные называют этот шифр ключевым автоключом (Key Auto-Key, KAK). При шифровании генератор потока ключей один за другим выдает биты потока ключей. При дешифрировании другой генератор потока ключей один за другим выдает идентичные биты потока ключей. Это работает, если оба генератора синхронизированы . Если один из них пропускает один из циклов, или если бит шифротекста теряется при передаче , то после ошибки каждый символ шифротекста будет расшифрован неправильно .

Положительная сторона синхронных фильтров - это отсутствие распространения ошибок . Если при передаче бит изменит свое значение, что намного вероятнее его потери, то только испорченный бит будет дешифрован неправильно. Все предшествующие и последующие биты не изменятся . Генератор должен выдавать один и тот же поток ключей и для шифрования, и для дешифрирования, следовательно, выход генератора должен быть предопределен. Если он реализуется на конечном автомате (т.е., компьютере), последовательность со временем повторится. Такие генераторы потока ключей называются периодическими. За исключением одноразовых блокнотов все генераторы потока ключей являются периодическими . Конкретная длина периода зависит от приложения. Синхронные потоковые шифры также предохраняют от любых вставок и удалений шифротекста , так как они приводят к потере синхронизации и будут немедленно обнаружены . Однако, они не защищают полностью от битовых сбоев. Синхронные потоковые шифры чувствительны к **вскрытию вставкой.**

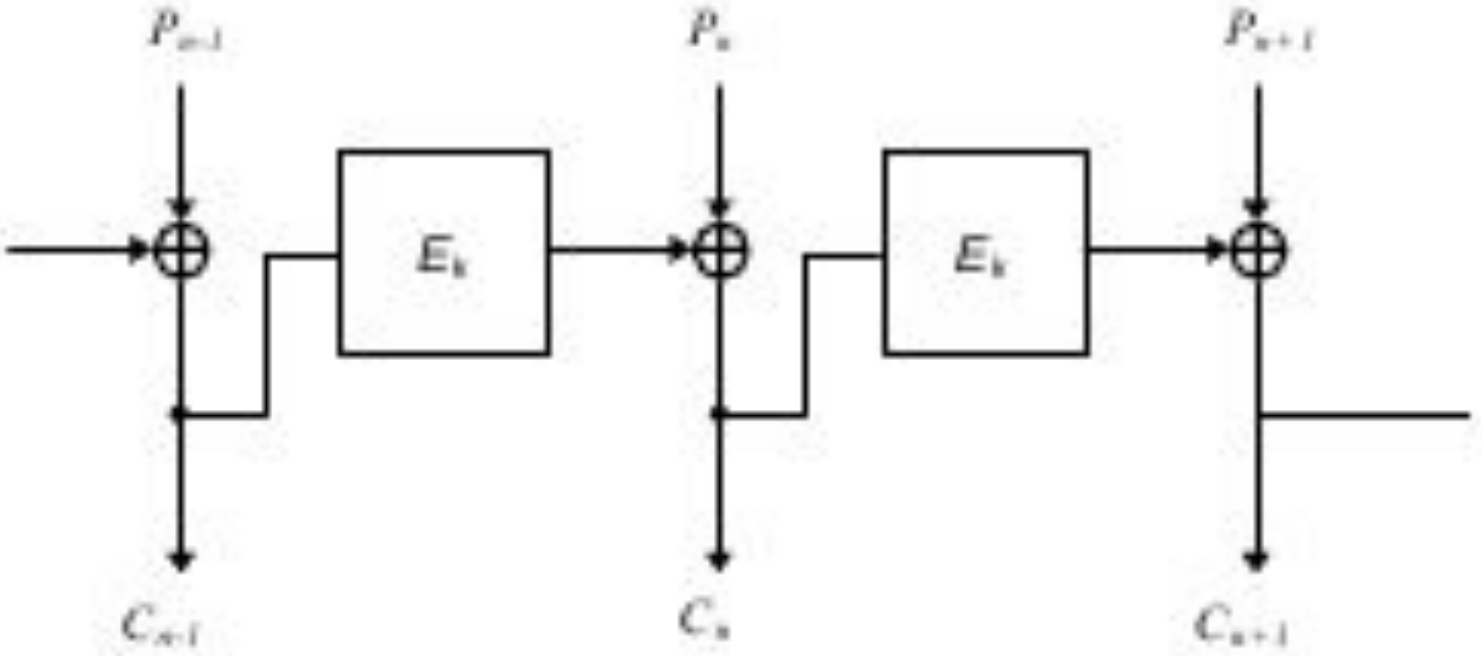
Блочный шифр также может быть реализованы как самосинхронизирующийся потоковый шифр, такой режим называется **режимом обратной связи по шифру** (cipher-feedback, **CFB**). В режиме CBC шифрование не могло начаться, пока не получен целый блок данных. Это создает проблемы для некоторых сетевых приложений . Например, в безопасной сетевой среде терминал должен иметь возможность передавать главному компьютеру каждый символ сразу, как только он введен. Если данные нужно обрабатывать байтами, режим CBC также не работает. В режиме CFB единица зашифрованных данных может быть меньше размера блока . Вы можете шифровать данные по одному биту с помощью 1-битового CFB, хотя использование для единственного бита полного шифрования блочным шифром потребует много ресурсов, потоковый шифр в этом случае был бы лучше. (Уменьшение количества циклов блочного фильтра для повышения скорости не рекомендуется.) Можно также использовать 64-битовый CFB, или любой n-битовый CFB, где n больше или равно размеру блока.

Если размер блока алгоритма - n, то -битовый CFB выглядит следующим образом: Ci = Pi ⊕Ek(C_{i-1})

P i = C i ⊕E k(C . .)



Режим 8-битовой обратной связи по шифру.



n-битовый CBF с n-битовым алгоритмом.

Как и режим CBC, режим CFB связывает вместе символы открытого текста так, что шифротекст зависит от всего предшествующего открытого текста.

Вектор инициализации Для инициализации процесса CFB в качестве входного блока алгоритма может использоваться вектор инициализации IV. Как и в режиме CBC IV не нужно хранить в секрете, но он должен быть уникальным.

Если IV в режиме CFB не уникален, криптоаналитик может раскрыть соответствующий открытый текст. IV должен меняться для каждого сообщения. Это может быть последовательный номер, увеличивающийся для каждого нового сообщения и не повторяющийся в течение времени жизни ключа . Если данные шифруются с целью последующего хранения, IV может быть функцией индекса, используемого для поиска данных.

В режиме CFB **ошибка** в открытом тексте влияет на весь последующий шифротекст, но самоустраняется при дешифрировании. В общем случай в n-битовом режиме CFB одна ошибка шифротекста влияет на дешифрирование текущего и следующих m/n-l блоков, где m - размер блока. CFB самовосстанавливается и после ошибок синхронизации.

Режим выходной обратной связи (Output-feedback, **OFB**) представляет собой метод использования блочного шифра в качестве синхронного потокового шифра. Этот режим похож на CFB за исключением того, что n битов предыдущего выходного блока сдвигаются в крайние правые позиции очереди. Дешифрирование является обратным процессом. Такой режим называется n-битовым OFB. И при шифровании, и при дешифриро- вании блочный алгоритм работает в режиме шифрования. Это иногда называют внутренней обратной связью, потому что механизм обратной связи не зависит ни от потоков открытого текста, ни от потоков шифротекста. Если размер блока алгоритма n, то n-битовый алгоритм OFB выглядит, как показано на :

$Ci = Pi \oplus Si$; $Si = Ek *(S\ i-1)$

$Pi = Ci \oplus Si$; $Si = Ek *(S_{i-1})$, где Si –состояние, не зависящее ни от открытого текста, ни от шифротекста . Положительные свойства OFB - большая часть работы может быть выполнена автономно, даже до того, как появится открытый текст сообщения. Когда сообщение появится, для получения шифротекста над сообщением и выходом алгоритма нужно будет выполнить операцию XOR.

Потеря синхронизации смертельна. Если сдвиговые регистры при шифровании и при дешифрировании отличаются, то восстановленный открытый текст- бессмыслица . Любая система, использующая режим OFB, должна включать механизм обнаружения потери синхронизации и механизм заполнения обоих сдвиговых регистров новым (или одинаковым) IV для восстановления синхронизации. OFB стоит использовать только, когда размер обратной связи совпадает с размером блока.

Блочные шифры в режиме счетчика используют в качестве входов алгоритма последовательные номера. Для заполнения регистра используется счетчик, а не выход алгоритма шифрования . После шифрования каждого блока счетчик инкрементируется на определенную константу, обычно единицу . Для этого режима свойства синхронизации и распространения ошибки такие же, как и для OFB. Режим счетчика решает проблему n-битового выхода режима OFB, где n меньше длины блока. К счетчику не предъявляется никаких особых требований, он не должен проходить по порядку все возможные значения. В качестве входа блочного алгоритма можно использовать генераторы случайных чисел

Потоковый шифр в режиме счетчика может генерировать i-ый бит, ki, без выдачи всех предшествующих ключевых битов. Устанавливается счетчик вручную в i-ое внутреннее состояние и генерируется бит. Это полезно для закрытия файлов данных с произвольным доступом, можно расшифровать конкретный блок данных не расшифровывая целый файл.

Другие режимы

Для использования блочного алгоритма в **режиме сцепления блоков** (block chaining, **BC**), выполняется XOR входа блочного шифра и результата XOR всех предыдущих блоков шифротекста. Как и для CBC используется IV. Математически это выглядит как: $C_i = E_k(P_i + F_i)$; $F_{i+1} = F_i + C_i$; $P_i = F_i + D_k(C_i)$; $F_{i+1} = F_i + C_i$

Как и CBC, обратная связь процесса BC приводит к распространению ошибки в открытом тексте.

-Режим сцепления блоков шифра с распространением ошибки (propagating cipher block chaining, PCBC)

- Сцепление блоков шифра с контрольной суммой (cipher block chaining with checksum, CBCC) представляет собой вариант CBC

-Выходная обратная связь с нелинейной функцией (output feedback with a nonlinear function, OFBNLF) представляет собой вариант и OFB, и ECB.

-Сцепление блоков открытого текста (plaintext block chaining, PBC) похоже на CBC за исключением того, что операция XOR выполняется для с блока открытым текстом и для предыдущего блока открытого текста, а не блока шифротекста.

-Обратная связь по открытому тексту (plaintext feedback, PFB) похожа на CFB за исключением того, что для обратной связи используется не шифротекст, а открытый текст.

-Существует также сцепление блоков шифротекста по различиям открытого текста (cipher block chaining of plaintext difference, CBCPD)