ЛЕКЦИЯ 4.
Классические методы шифрования (ч.2)

4.1. Методы перестановки.

4.2. Блочные шифры.

4.3. Режимы применения блочных шифров.

4.4. Композиции (комбинации) шифров.

Применение перестановок

Шифры, созданные с помощью перестановок, называют **перестановочными шифрами**.

Простейший из таких шифров использует **преобразование «лесенки».**

Например, для шифрования сообщения ***«meet me after the toga party»*** по **методу лесенки** со ступеньками ***длиной 2***, оно записывается в виде:

m e m a t r h t g p r y
e t e f e t e o a a t

Шифрованное сообщение:

MEMATRHTGPRYETEFETEOAAT

Более сложная схема:

- Запись текста сообщения в горизонтальные строки одинаковой длины и последующее считывание текста ***столбец за столбцом***, но не по порядку, а в соответствии с некоторой *перестановкой* столбцов.
- Порядок считывания столбцов при этом становится ***ключом*** алгоритма.

Пример.

Ключ: **3 4 2 1 5 6 7**

Открытый текст: a t t a c k p
o s t p o n e
d u n t i l t
w o a mx y z

Шифрованный текст: **TTNAAPTMTSUOAODWCOIXKNLYPETZ**

Перестановочный шифр можно сделать ***существенно более защищенным***, выполнив шифрование с использованием перестановок несколько раз.

Перестановка, основанная на **кубике Рубика (объемная (многомерная) перестановка):**

1.Открытый текст записывается в ячейки ***граней куба*** по строкам.
2.После осуществления заданного числа заданных поворотов слоев куба считывание шифртекста осуществляется по столбикам.
3. Сложность расшифрования в этом случае определяется ***количеством ячеек*** на гранях куба и сложностью ***выполненных поворотов*** слоев.

Секретная система **"Рубикон“**

1. В качестве **пространственных многомерных структур**, на основании объемных преобразований которых осуществляются перестановки, в системе "***Рубикон***" используются **трехмерные куб** и **тетраэдр**.
2. Генерируется **уникальная версия** *алгоритма* и **ключа** криптографических преобразований на основании некоторого ***секретного параметра (пароля).***
3. **Криптостойкость** данной системы определяется
   а) ***длиной*** ключа,
   б) криптостойкостью отдельных ***функциональных элементов*** алгоритма криптографических преобразований,
   в) ***количеством*** таких преобразований.

Блочные шифры

Блочными называются шифры, в которых логической единицей шифрования является некоторый блок открытого текста, после преобразований которого получается блок шифрованного текста такой же длины.

Обычно используются блоки размером 64 бита.

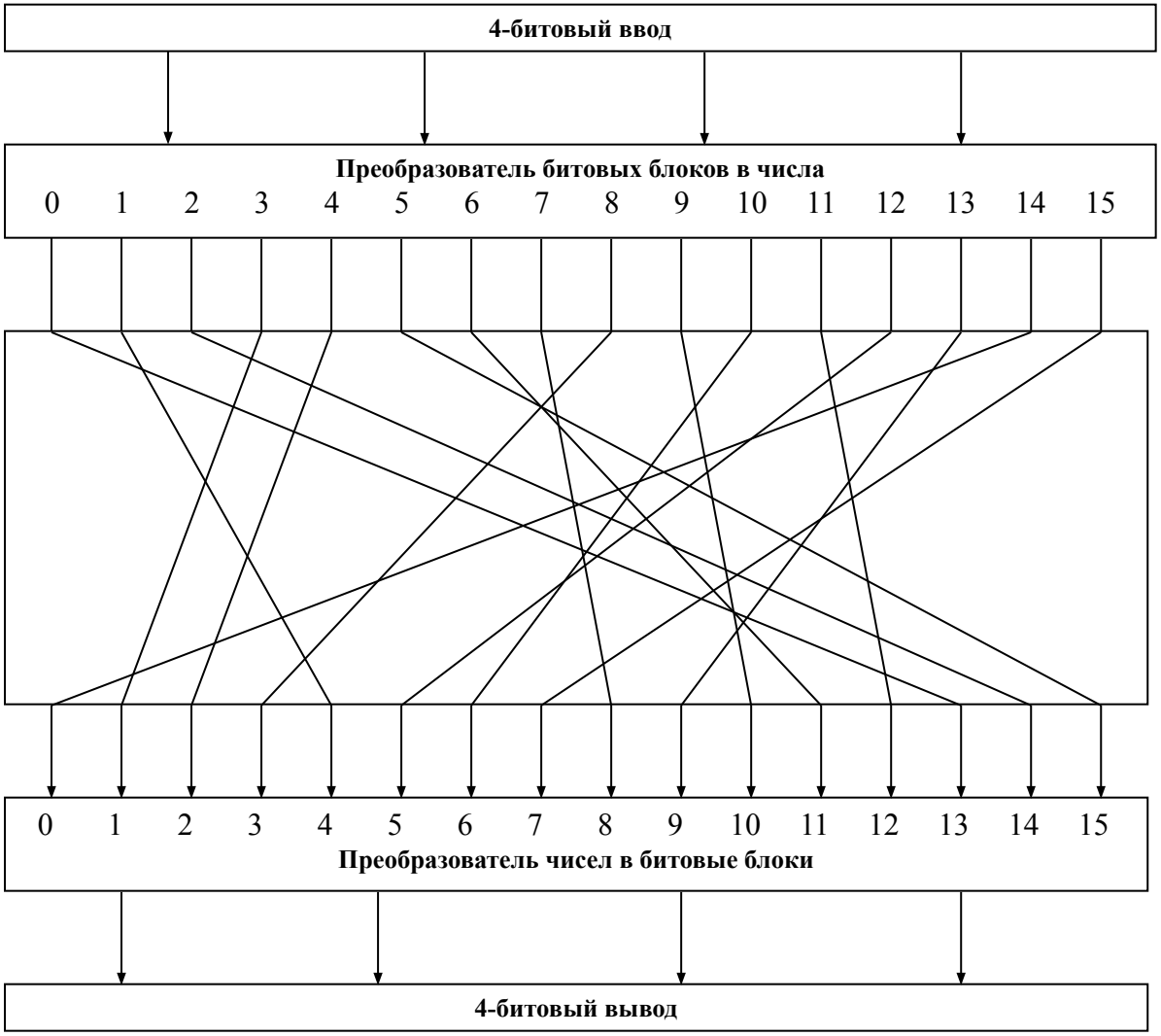
Чтобы шифрование было ***обратимым*** (т.е. чтобы обеспечивалась возможность *дешифрования*), каждый из таких блоков должен преобразовываться в свой ***уникальный*** блок **шифрованного текста**.

Такие преобразования называются **обратимыми**, или несингулярными.

Примеры несингулярного и сингулярного преобразований для **n = 2**.

Открытый текст	**Шифрованный текст**	**Открытый текст**	**Шифрованный текст**
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

Общая схема поблочной подстановки n-битовых блоков (случай n=4)



Таблицы шифрования и дешифрования для блочного шифра

Открытый текст	Шифрованный текст	Открытый текст	Шифрованный текст
0000	1110	0000	1110
0001	0100	0001	0011
0010	1101	0010	0100
0011	0001	0011	1000
0100	0010	0100	0001
0101	1111	0101	1100
0110	1011	0110	1010
0111	1000	0111	1111
1000	0011	1000	0111
1001	1010	1001	1101
1010	0110	1010	1001
1011	1100	1011	0110
1100	0101	1100	1011
1101	1001	1101	0010
1110	0000	1110	0000
1111	0111	1111	0101

Диффузия и конфузия

Идеальный шифр — шифр, который полностью скрывает в шифрованном тексте все ***статистические закономерности*** открытого текста.

Диффузия - ***рассеяние*** статистических особенностей открытого текста по широкому диапазону статистических характеристик шифрованного текста.

Это достигается тем, что значение *каждого элемента* **открытого** текста влияет на значения многих элементов **шифрованного** текста или, что эквивалентно, любой из элементов **шифрованного** текста зависит от *множества элементов* **открытого** текста.

Пример

Шифрование сообщения $M = m_1, m_2, m_3, \ldots$

с помощью операции ***усреднения***

$$y_n = \sum_{i=1}^{k} m_{n+i} \pmod{26},$$

когда для получения буквы y_n шифрованного текста складываются k последовательных букв открытого текста.

Конфузия - усложнение в максимальной степени статистической взаимосвязи *между шифрованным текстом и ключом* (опять же с целью противостояния попыткам определения ключа).

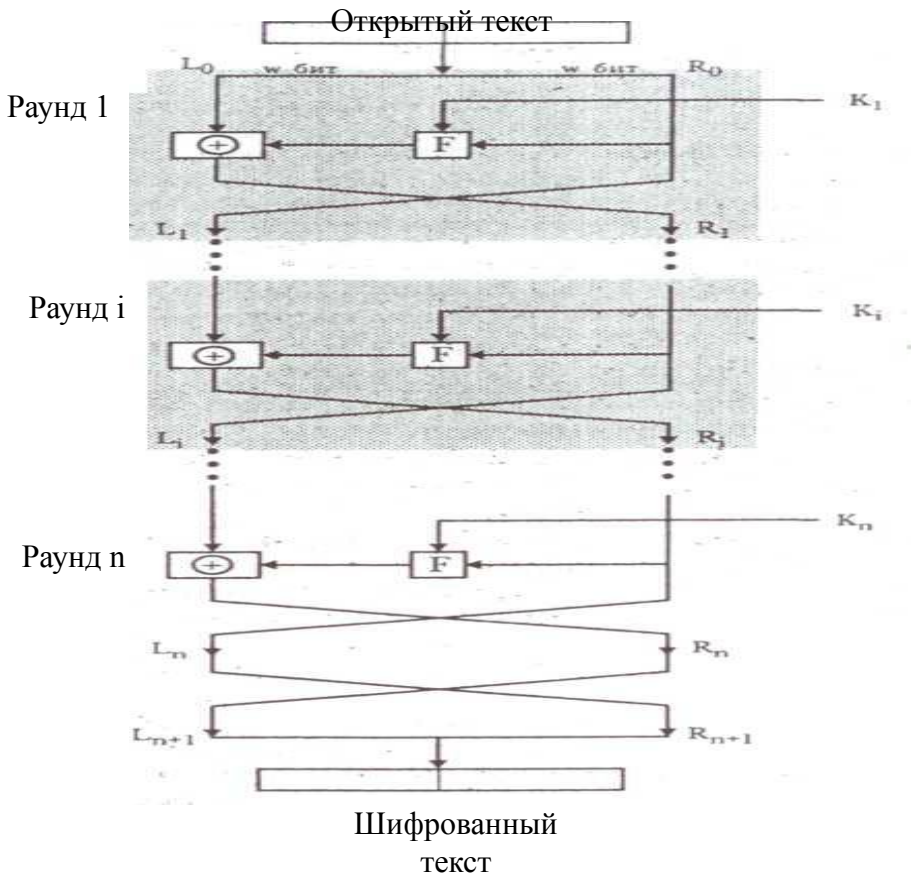
Это достигается использованием ***сложных*** подстановочных алгоритмов: простые **линейные** подстановочные функции увеличивают беспорядок лишь в незначительной степени.

Шифр Файстеля

ИДЕЯ: Аппроксимация **подстановочного** шифра **продукционными шифрами**, которые строятся на ***последовательном применении*** как *минимум двух* базовых шифров с тем, чтобы полученный результат обладал более высокой стойкостью по сравнению с любым из этих базовых шифров в отдельности.

Структура шифра Файстеля

(является частным случаем так называемой ***подстановочно-перестановочной схемы (substitution-permutation network),*** предложенной ***Шенноном***).



Описание алгоритма Файстеля

1. На вход алгоритма шифрования подается блок **открытого** текста длиной ***2w*** битов и ключ ***K***.
2. Блок открытого текста разделяется на ***две равные*** части - $\mathbf{L_0}$ и $\mathbf{R_0}$, которые последовательно проходят через ***n* раундов обработки**,
3. после чего ***объединяются*** снова для получения блока шифрованного текста соответствующей длины.

Схема раунда обработки

Для раунда ***i*** в качестве **входных данных** выступают $\mathbf{L_{i-1}}$ и $\mathbf{R_{i-1}}$, полученные на выходе предыдущего, ***(i − 1)* −го,** раунда, и подключ $\mathbf{K_i}$, вычисляемый по общему ключу ***K***. Как правило, подключи $\mathbf{K_i}$ *отличаются* как от общего ключа **K**, так и друг от друга.

1. Сначала для *левой* половины блока данных выполняется **операция подстановки:** применение к *правой* половине блока данных некоторой ***функции раунда* F** и последующее сложение полученного результата с *левой* половиной блока данных с помощью операции ***XOR*** (**исключающего "ИЛИ"**). *Для всех раундов* ***функция раунда*** *имеет* одну и ту же структуру, *но зависит от параметра — подключа раунда* $\boldsymbol{K_i}$.
2. После подстановки выполняется **перестановка**, представляющая собой ***обмен местами*** двух половин блока данных.

Конкретная реализация схемы ***Файстеля***
определяется выбором значений следующих **параметров**

* **Размер блока.** Чем ***больше*** размер блока, тем выше *надежность* шифра (при прочих равных условиях), но тем ниже скорость выполнения операций шифрования и дешифрования. *Разумный компромисс - блок размером* ***64 бита***.

* **Размер ключа.** Чем ***длиннее*** ключ, тем выше *надежность* шифра, но большая длина ключа тоже может быть причиной слишком *медленного выполнения* операций шифрования и дешифрования. *На сегодняшний день используются* ***128-битовые ключи***.

* **Число раундов обработки.** За один раунд обработки данных обеспечивается *недостаточно высокая* надежность, но уровень надежности шифра ***повышается*** с каждым новым раундом обработки. *Как правило, число раундов выбирают равным* **16**.

* **Алгоритм вычисления подключей.** Чем ***сложнее*** этот алгоритм, тем в большей степени затрудняется криптоанализ шифра.

* **Функция раунда.** ***F-функция*** представляет собой последовательность *зависящих от ключа* **нелинейных замен, перемешивающих перестановок и сдвигов.** ***Усложнение*** функции, как правило, ведет к *повышению* стойкости шифра с точки зрения криптоанализа.

* **Скорость выполнения программ шифрования/дешифрования.** Скорость программного выполнения алгоритма оказывается важным фактором, когда функции шифрования **встраиваются** в приложения или утилиты .

* **Простота анализа.** Если алгоритм прост и понятен, его легче анализировать с точки зрения **уязвимости для криптоанализа** и, следовательно, легче совершенствовать с целью достижения более высокого уровня надежности.

Алгоритм дешифрования шифра Файстеля

Применяется **тот же алгоритм**, но:

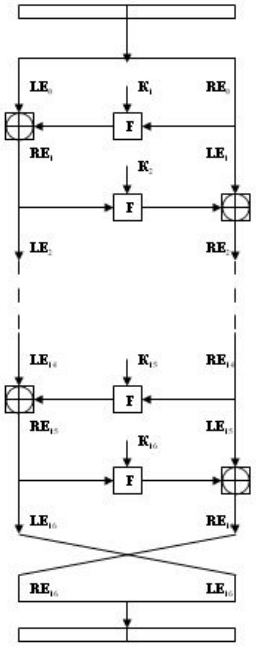
1. на вход подается ***шифрованный*** текст,
2. подключи – K_i, используются в обратной последовательности:

для ***первого*** раунда берется подключ $\mathbf{K_n}$,
для ***второго*** — $\mathbf{K_{n-1}}$, и так далее,
пока не будет введен ключ $\mathbf{K_1}$ для ***последнего*** раунда.

Данные, обрабатываемые **алгоритмом шифрования,** - $\mathbf{LE_i}$ и $\mathbf{RE_i}$,
данные, обрабатываемые ***алгоритмом дешифрования***, - $\mathbf{LD_i}$ и $\mathbf{RD_i}$ **,**
выходное значение i-го раунда шифрования - $\mathbf{LE_i} \,||\, \mathbf{RE_i}$ (*конкатенация* $\mathbf{L_i}$ и $\mathbf{R_i}$), ***входное*** значение для ***(16 – i)*-го** раунда дешифрования - $\mathbf{LD_i} \,||\, \mathbf{RD_i}$.

Шифрование

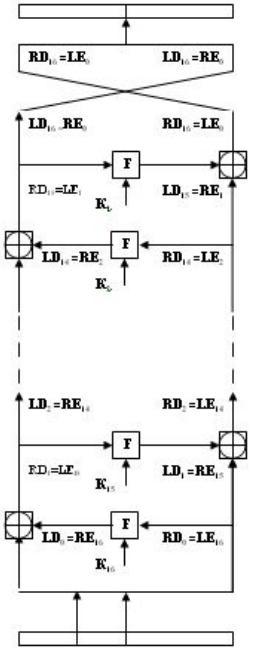
Ввод (открытый текст)



Вывод (шифрованный текст)

дешифрование

Вывод (открытый текст)



Ввод (*шифрованный* текст)

На **каждой стадии** процесса ***дешифрования*** очередное промежуточное значение с точностью до перестановки половинок этого значения равно промежуточному значению, получаемому на соответствующей стадии процесса ***шифрования***.

На входе **первого раунда** алгоритма - $RE_{16}||LE_{16}$, равное значению результата **16-го раунда** шифрования с переставленными 32-битовыми половинами.

Процесс шифрования:

$$LE_{16}=RE_{15},$$

$$RE_{16}=LE_{15} \oplus F(RE_{15},K_{16}).$$

*Операция **XOR** (исключающее «ИЛИ») имеет следующие свойства:*

$$[A \oplus B] \oplus C = A \oplus [B \oplus C],$$
$$D \oplus D=0,$$
$$E \oplus 0 = E.$$

*Процесс **дешифрования*** :

$$LD_1=RD_0=LE_{16}=RE_{15},$$

$$RD_1=LD_0 \oplus F(RD_0,K_{16})=RE_{16} \oplus F(RE_{15},K_{16})=[LE_{15} \oplus F(RE_{15},K_{16})] \oplus F(RE_{15},K_{16}).$$

Для ***i*-той итерации** алгоритма **шифрования** имеем

$$LE_i = RE_{i-1},$$

$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i).$$

Преобразуя, получим: $RE_{i-1} = LE_i,$

$$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i).$$

Т.е. входные данные для i-й итерации выражаются в виде функций от выходных данных.

На выходе ***последнего*** раунда процесса дешифрования формируется значение

$$\mathbf{RE_0 \,||\, LE_0}.$$

Для восстановления исходного открытого текста остается лишь ***32-битовый обмен***, что завершает доказательство пригодности схемы ***Файстеля*** для процесса дешифрования.

Достоинства *шифров Файстеля*:

• процедуры шифрования и расшифрования совпадают, с тем исключением, что ключевая информация при расшифровании используется в *обратном* порядке;

• для построения устройств шифрования можно использовать те же блоки в цепях шифрования и расшифрования.

Недостаток

На каждой итерации изменяется только *половина блока* обрабатываемого текста, что приводит к необходимости **увеличивать число итераций** для достижения требуемой стойкости.

Режимы применения блочных шифров

Режим **электронной кодировочной книги (*ECB – Electronic Code Book*) -** каждый блок исходного текста шифруется блочным шифром независимо от других.

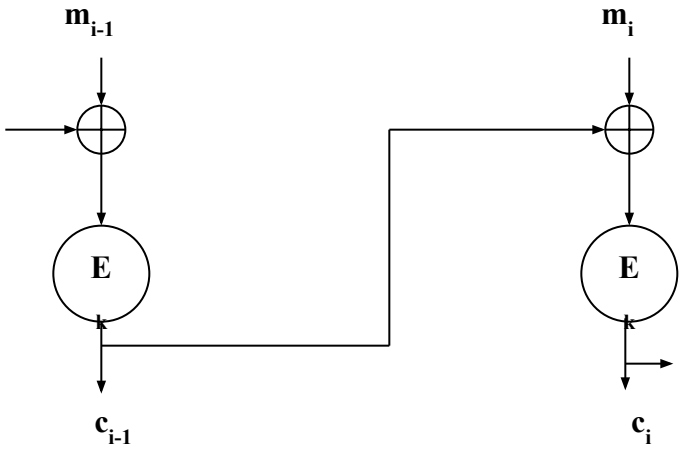
1. Стойкость режима **ECB** равна стойкости *самого шифра.*
Однако, структура исходного текста при этом не скрывается. Каждый одинаковый блок исходного текста приводит к появлению одинакового блока шифрованного текста.
Исходным текстом можно *легко манипулировать* путем **удаления, повторения или перестановки блоков.**
2. Скорость шифрования равна скорости блочного шифра.
Режим **ECB** допускает **простое распараллеливание** для увеличения скорости шифрования, но никакая обработка невозможна до поступления блока (за исключением ***генерации ключей***).

$$c_i = E_k(m_i);\ m_i = D_k(c_i)$$

Режим **электронной кодировочной книги (*ECB – Electronic Code Book*)**

Режим **ECB** соответствует ***режиму простой замены***, определенному в ГОСТе.

Режим **сцепления блоков шифрованного текста** (***CBC– Cipher Block Chaining***) -каждый блок исходного текста складывается поразрядно по модулю 2 с предыдущим блоком шифрованного текста, а затем шифруется .
Для начала процесса шифрования используется *синхропосылка* (или *начальный вектор*), которая передается в канал связи в **открытом** виде.



$$c_i = E_k(m_i \oplus c_{i-1});\ m_i = D_k(c_i) \oplus c_{i-1}$$

Режим **сцепления блоков шифрованного текста** (***CBC – Cipher Block Chaining***)

1. Стойкость режима **CBC** равна стойкости блочного шифра, лежащего в его основе. Структура исходного текста **скрывается** за счет сложения предыдущего блока шифрованного текста с очередным блоком открытого текста. Стойкость **увеличивается**, т.к. становится невозможной прямая манипуляция исходным текстом.
2. Скорость шифрования равна скорости работы блочного шифра, но простого способа ***распараллеливания*** процесса шифрования **не существует**, хотя расшифрование может проводиться параллельно.

Модификации **режима CBC:**

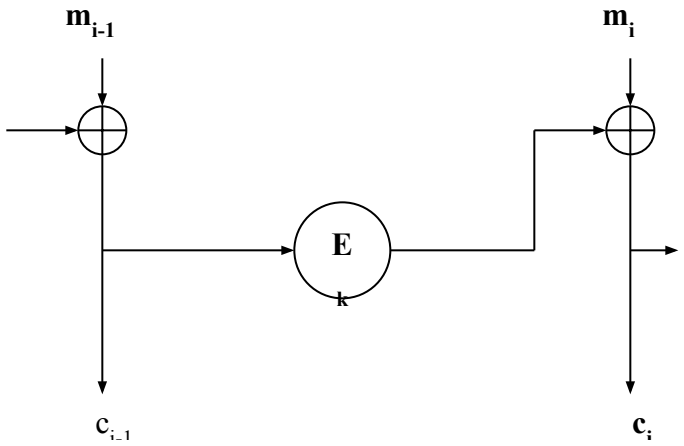
− Режим **сцепления блоков шифрованного текста с распространением** (***PCBC – Propagating CBC***) **.**
Отличается тем, что по модулю 2 складывается как предыдущий блок **шифрованного**, так и ***исходного*** текста:

$$c_i = E_k(m_i \oplus c_{i-1} \oplus m_{i-1}),$$

$$m_i = c_{i-1} \oplus m_{i-1} \oplus D_k(c_i).$$

− Режим **сцепления блоков шифрованного текста с контрольной суммой** (***CBCC – CBC with Checksum***) .
Отличается тем, что к ***последнему*** блоку исходного текста перед шифрованием прибавляется сумма по модулю два ***всех предыдущих блоков*** исходного текста. Это дает возможность проконтролировать целостность передаваемого текста с небольшими дополнительными накладными расходами.

Режим **обратной связи по шифрованному тексту** (***CFB – Cipher Feedback***) - предыдущий блок шифрованного текста шифруется еще раз, и для получения очередного блока шифрованного текста результат складывается ***поразрядно по модулю 2*** с блоком исходного текста. Для начала процесса шифрования также используется *начальный вектор*.



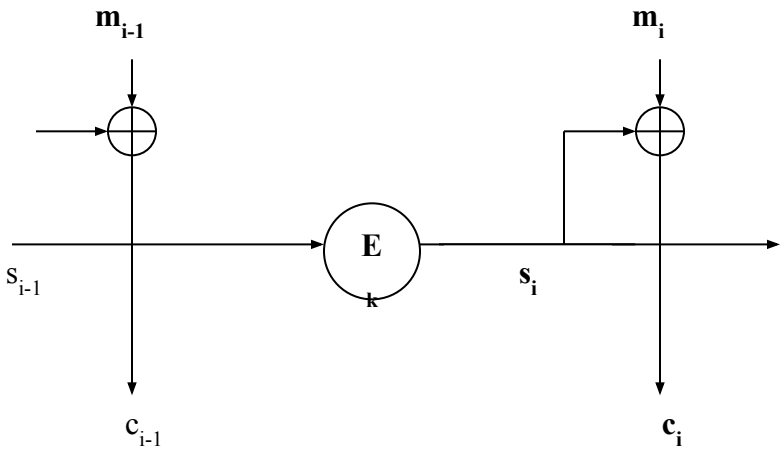
$$c_i = m_i \oplus E_k(c_{i-1}); \; m_i = E_k(c_{i-1}) \oplus c_i$$

Режим **обратной связи по шифрованному тексту** (***CFB – Cipher Feedback***)

1. Стойкость режима **CFB** равна стойкости блочного шифра, лежащего в его основе. Структура исходного текста скрывается за счет использования операции сложения по модулю 2. Манипулирование исходным текстом путем удаления блоков из начала или конца шифрованного текста становится ***невозможным***.

2. Скорость шифрования равна скорости работы блочного шифра. Простого способа ***распараллеливания*** процесса шифрования не существует. Этот режим в точности соответствует **режиму гаммирования с обратной связью** алгоритма ГОСТ 28147–89.

Режим **обратной связи по выходу (*OFB– Output Feedback*)** - подобен режиму **CFB** за исключением того, что величины, складываемые по модулю 2 с блоками исходного текста, генерируются ***независимо*** от исходного или шифрованного текста.
Для начала процесса шифрования также используется *начальный вектор*.



$$c_i = m_i \oplus s_i;\ m_i = c_i \oplus s_i;\ s_i = E_k(s_{i-1}).$$

Режим **обратной связи по выходу (*OFB – Output Feedback*).**

Режим **OFB** обладает преимуществом перед режимом **CFB** – любые битовые ошибки, возникшие в процессе передачи, не влияют на расшифрование последующих блоков. Однако, возможна ***простая манипуляция*** исходным текстом путем изменения шифрованного текста.
Существует модификация этого режима под названием **«режим обратной связи по выходу с нелинейной функцией»** - в этом случае на каждом шаге меняется также и **ключ шифрования:**

$$k_i = E_k(k_{i-1}).$$

Композиции (комбинации) шифров

Наиболее часто применяемые комбинации:

− подстановка и гамма,
− перестановка и гамма,
− подстановка и перестановка,
− гамма и гамма.

Примеры.

1. Шифр ***Д. Френдберга*** **-** комбинирует **многоалфавитную подстановку** с **генератором псевдослучайных чисел** .
Особенность алгоритма - при большом объеме шифртекста частотные характеристики символов шифртекста близки к ***равномерному распределению*** независимо от содержания открытого текста.

2. Система ***ЛЮЦИФЕР*** - строится на базе ***блоков подстановки (S-блоков)*** и ***блоков перестановки (P-блоков).***
Блок подстановки включает линейные и нелинейные преобразования:
− первый преобразователь S-блока осуществляет развертку двоичного числа из ***n*** разрядов в число по основанию ***2n***;
− второй преобразователь осуществляет свертку этого числа.
Блок перестановки осуществляет преобразование ***n*** − разрядного входного числа в ***n*** − разрядное число.
Входные данные (открытый текст) последовательно проходят через чередующиеся слои **32-разрядных** P-блоков и **8-разрядных** S-блоков.

Программное шифрование данных в системе ***ЛЮЦИФЕР*** показало *низкую производительность*, поэтому P- и S-блоки были реализованы **аппаратно**, что позволило достичь скорости шифрования до **100 Кбайт/с.**