Лекция 4.
Режимы шифрования
Алгоритм шифрования ГОСТ 28147-89 (ГОСТ Р.34.12-2015)

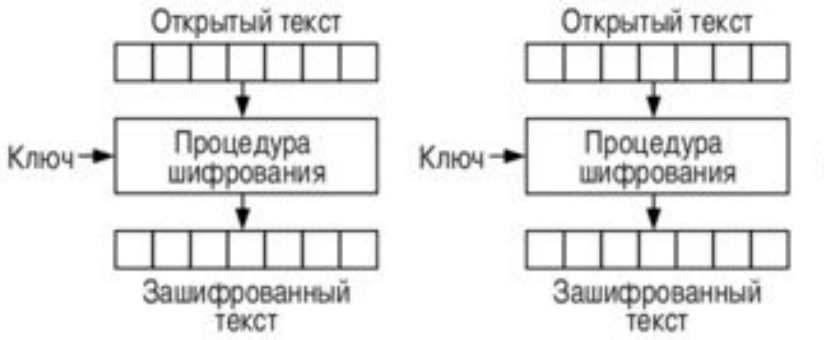
Режимы шифрования

- 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);

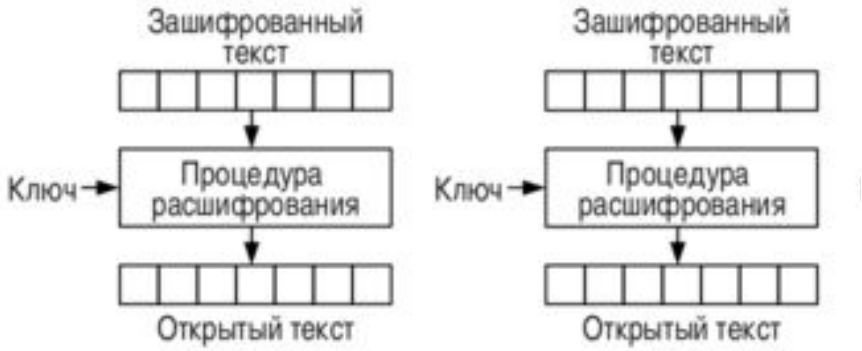
ЭЛЕКТРОННАЯ КОДОВАЯ КНИГА

- 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.

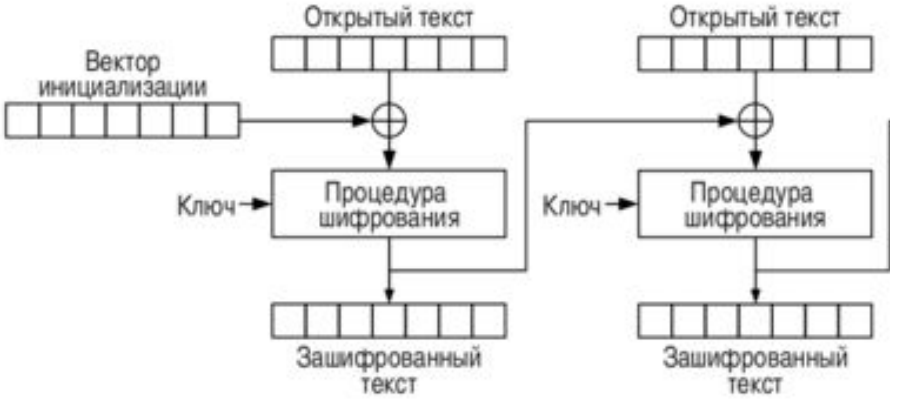
СЦЕПЛЕНИЕ БЛОКОВ ШИФРОТЕКСТА

- 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.

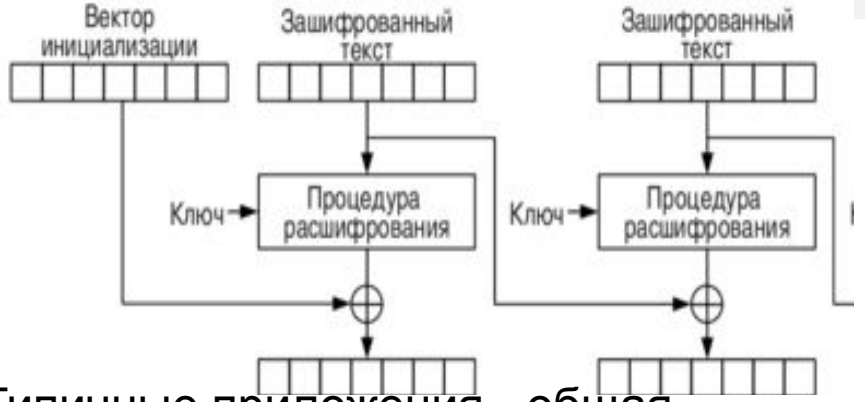
$C_0 = IV \qquad C_i = E_k(P_i \oplus C_{i-1})$

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

Шифрование в режиме CBC



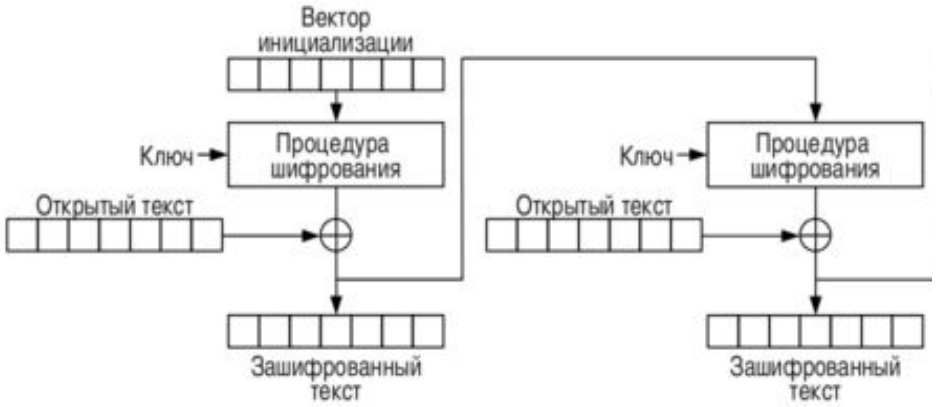
Расшифрование в режиме CBC



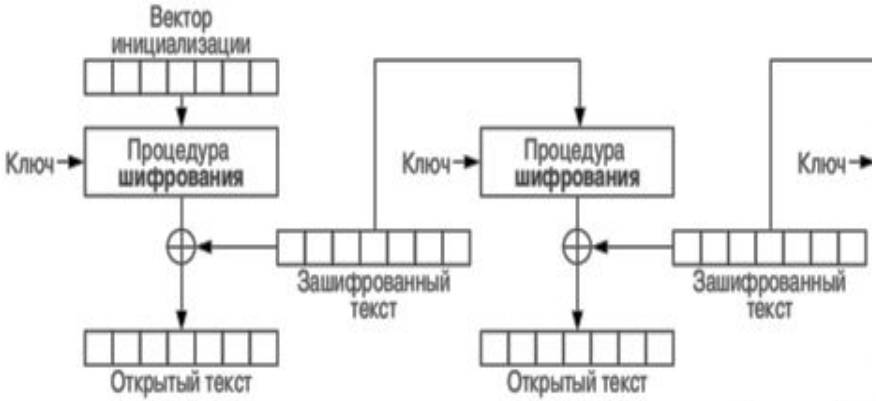
Типичные приложения - общая блокоориентированная передача, аутентификация.

ОБРАТНАЯ СВЯЗЬ ПО ШИФРОТЕКСТУ

- *Шифрование в режиме CFB*



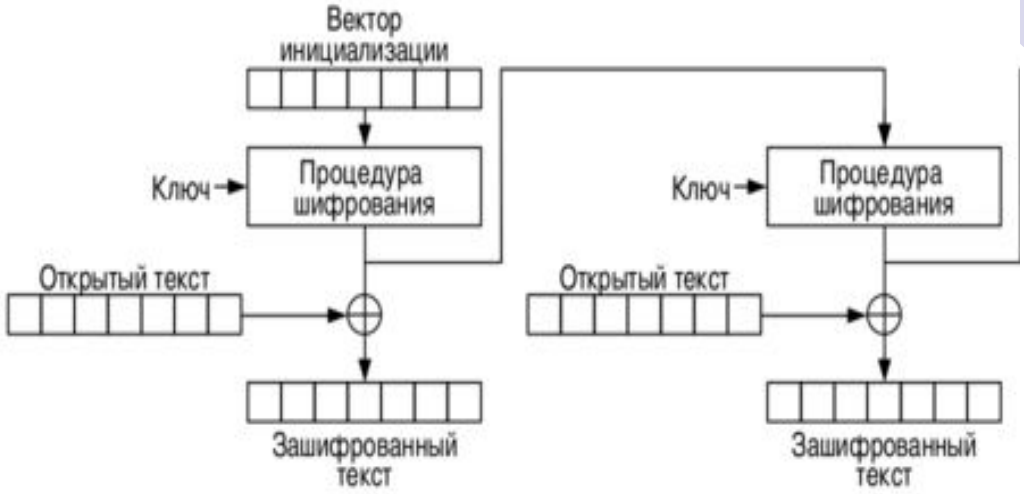
Расшифрование в режиме CFB



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.

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

ОБРАТНАЯ СВЯЗЬ ПО ВЫХОДУ



Сдвиговый регистр IV заполняется 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.

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

$$P_i = C_i \oplus O_i$$

ОБЩИЕ СВЕДЕНИЯ

- Полное название — «ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования».
- Размер блока - 64 бита
- Длина ключа - 256 бит
- Основа алгоритма — сеть Фейстеля с 32 раундами преобразования.

Элементы данных

- *X – элемент данных,* $|X|$ - размер X в битах. $0<\boldsymbol{X}<2^{|\boldsymbol{X}|}$.
- Состоит из элементов меньшего размера,

$X=(X_0,X_1,\ldots,X_{n-1})=X_0||X_1||\ldots||X_{n-1}$.

Размер $|X|=|X_0|+|X_1|+\ldots+|X_{n-1}|$.

- Интерпретируется как массив отдельных бит,

$X=(x_0,x_1,\ldots,x_{n-1})=x_0+2^1\cdot x_1+\ldots+2^{n-1}\cdot x_{n-1}$.

Для ГОСТа принята т.н. «little-endian» нумерация разрядов, т.е. внутри многоразрядных слов данных отдельные двоичные разряды с меньшими номерами являются менее значимыми.

Если над элементами данных выполняется логическая операция, то предполагается, что данная операция выполняется над соответствующими битами элементов.

$A \bullet B = (a_0 \bullet b_0, a_1 \bullet b_1, \ldots, a_{n-1} \bullet b_{n-1})$, где $n = |A| = |B|$, а символом «•» обозначается произвольная бинарная логическая операция; как правило, имеется в виду операция ***исключающего или***

КЛЮЧЕВАЯ ИНФОРМАЦИЯ

1. ***Ключ*** *K*- массив из восьми

32-битовых элементов кода, элементы ключа используются как 32-разрядные целые числа без знака: $\boldsymbol{K} = \{\boldsymbol{K_i}\}_{\mathbf{0 \le i \le 7}}$

Размер ключа $0 \le \boldsymbol{Ki} \le 2^{32}$ составляет $32 \cdot 8 = 256$ бит или 32 байта..

2. ***Таблица замен*** *H* - вектор, содержащий восемь ***узлов замены*** . Каждый узел замены является вектором, содержащим шестнадцать 4-битовых элементов замены, представляющих собой целые числа от 0 до 15, все элементы одного узла замены обязаны быть различными.

$$H = \{H_{i,j}\}_{0\le j\le 15}^{0\le i\le 7}, 0 \le H_{i,j} \le 15$$

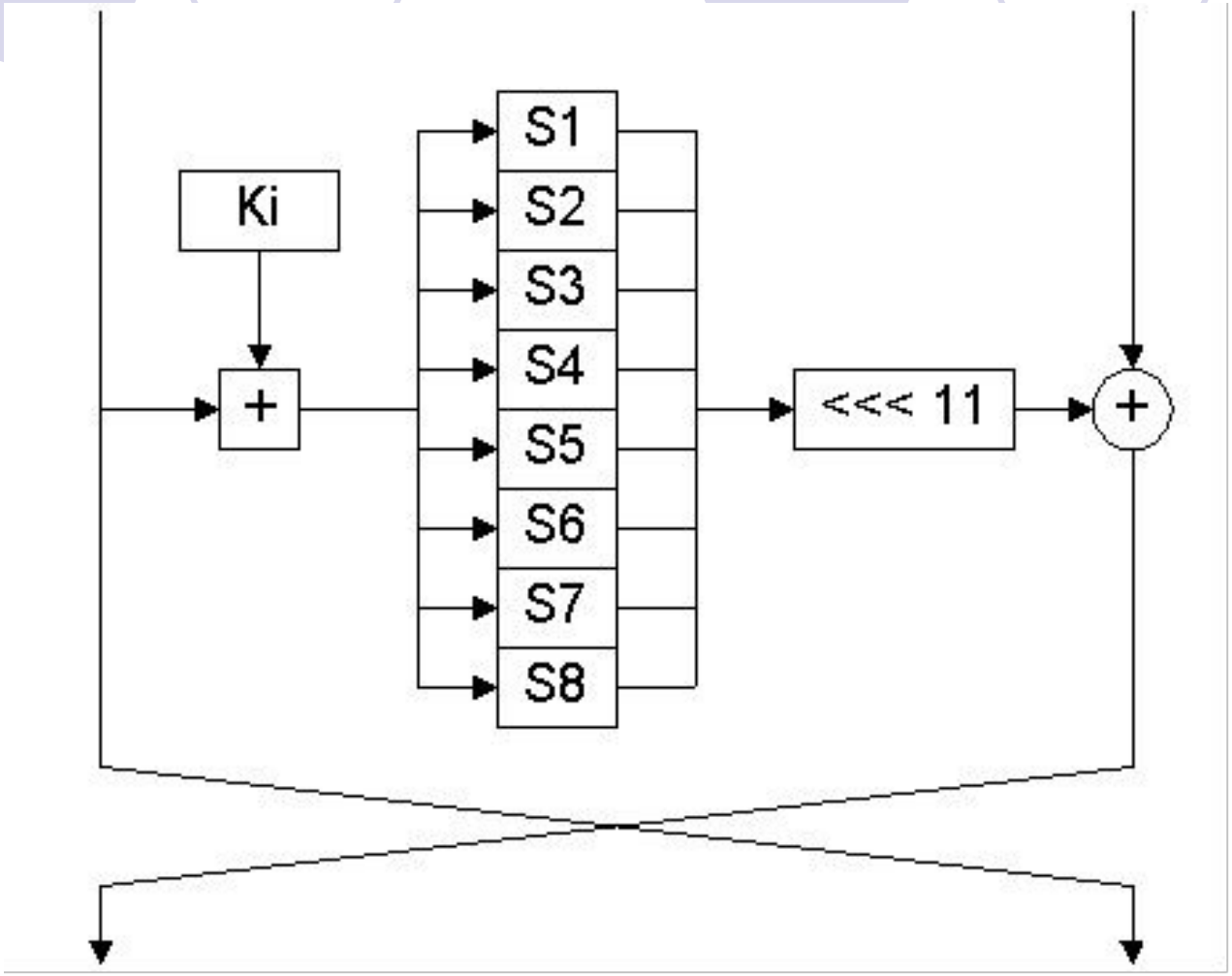
- общий объем таблицы замен: 8 узлов х 16 элементов/узел х 4 бита/элемент = 512 бит = 64 байта.

Принципы построения шифра ГОСТ

ГОСТ содержит алгоритмы трех уровней. В основе лежит «основной шаг», на базе которого строятся «базовые циклы», и уже на их основе построены практические процедуры шифрования. Специфика заключена именно в его основном шаге, На вход основного шага подается блок четного размера, старшая и младшая половины которого обрабатываются отдельно друг от друга. В ходе преобразования младшая половина блока помещается на место старшей, а старшая, скомбинированная с помощью операции побитового ***исключающего или*** с результатом вычисления некоторой функции, на место младшей.

Эта функция, принимающая в качестве аргумента младшую половину блока и некоторый элемент ключевой информации (*X*), является содержательной частью шифра и называется его ***функцией шифрования***. Соображения стойкости шифра требуют, чтобы размеры всех перечисленных элементов блоков были равны: |***N***1| = |***N***2| = |***X***|, в ГОСТ и DESe они равны 32 битам.

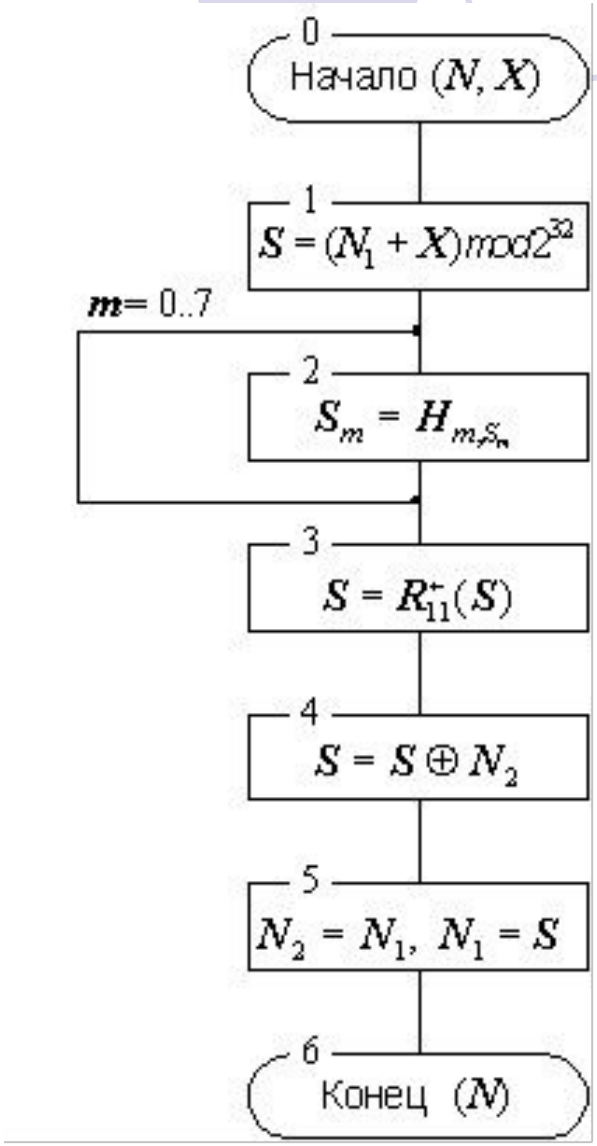
Основной шаг криптопреобразования



Пример таблицы замен

№S-блока	Значение															
1	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
2	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
3	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
4	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
5	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
6	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
7	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
8	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Основной шаг криптопреобразования



Шаг 0

N – преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака. *X* – 32-битовый элемент ключа;

Шаг 1

Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа;

Шаг 2

Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода: $S=(S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$, причем S_0 содержит 4 самых младших, а S_7 – 4 самых старших бита *S*.

Далее значение каждого из восьми блоков заменяется новым, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа.

Шаг 3

Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов.

Шаг 4

Побитовое сложение: значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Шаг 5

Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Шаг 6

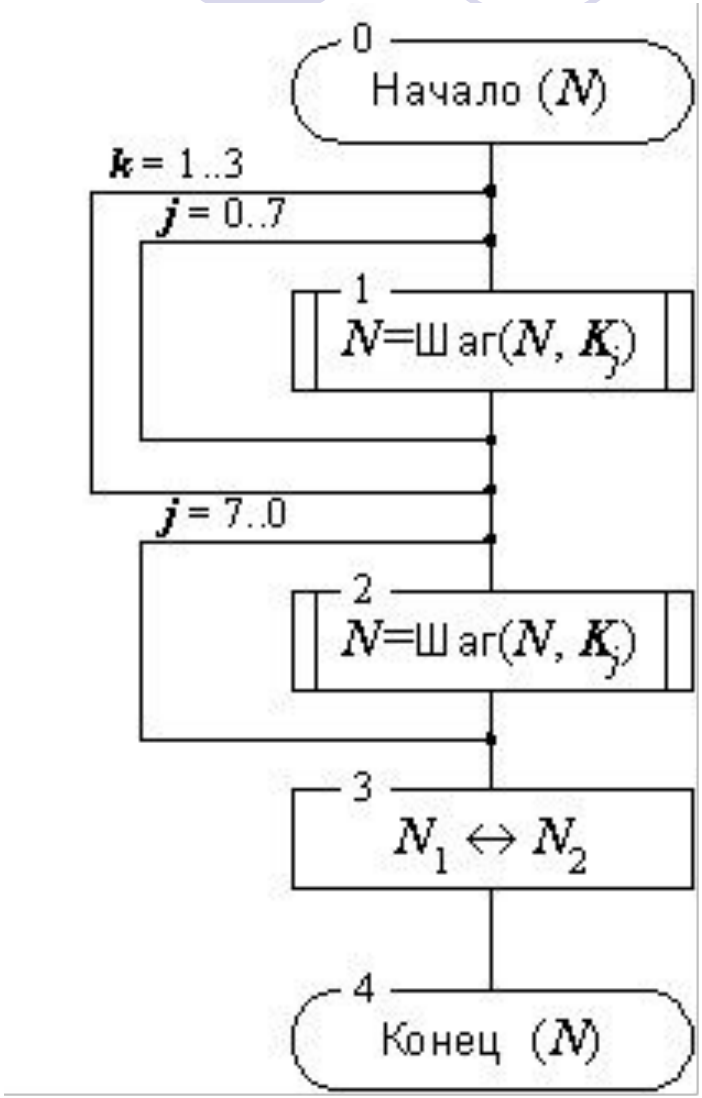
Полученное значение возвращается как результат выполнения алгоритма основного шага криптопреобразования.

Базовые циклы криптографических преобразований

Базовые циклы построены из ***основных шагов*** криптографического преобразования

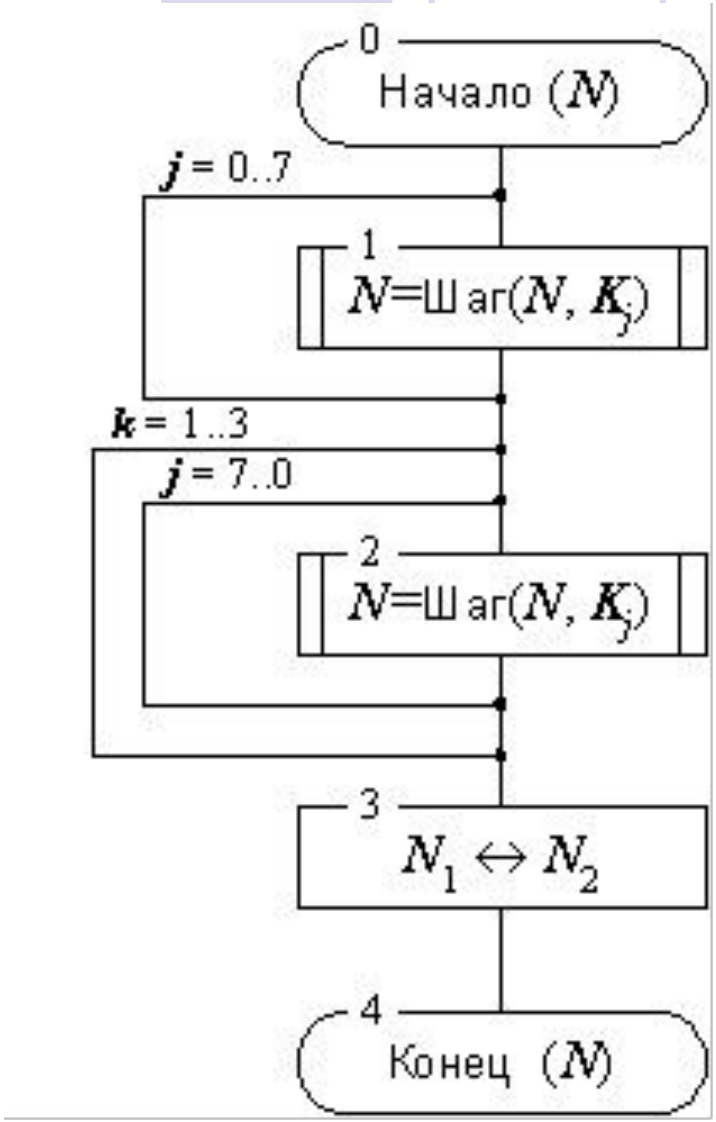
- цикл зашифрования (32-З);
- цикл расшифрования (32-Р);
- цикл выработки имитовставки (16-З).

Цикл зашифрования 32-3



$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

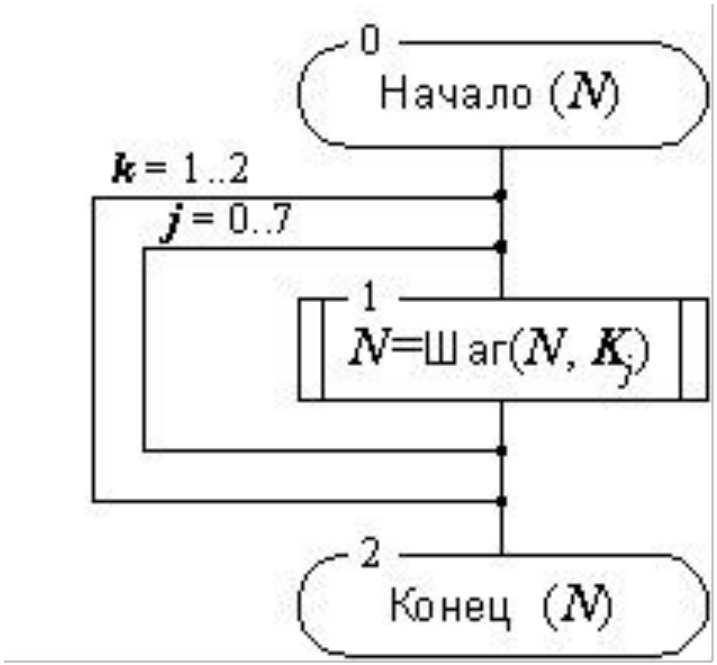
Цикл расшифрования 32-Р:



$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

Цикл выработки имитовставки 16-3:

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7.$



Цикл расшифрования должен быть обратным циклу зашифрования, то есть последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок: $Ц_{32\text{-}Р}(Ц_{32\text{-}З}(T))=T$, где T – произвольный 64-битовый блок данных, $Ц_{X}(T)$ – результат выполнения цикла X над блоком данных T. Для выполнения этого условия для алгоритмов, подобных ГОСТу, необходимо и достаточно, чтобы порядок использования ключевых элементов соответствующими циклами был взаимно обратным.

Из двух взаимно обратных циклов любой может быть использован для зашифрования, тогда второй должен быть использован для расшифрования данных, однако стандарт ГОСТ28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе.

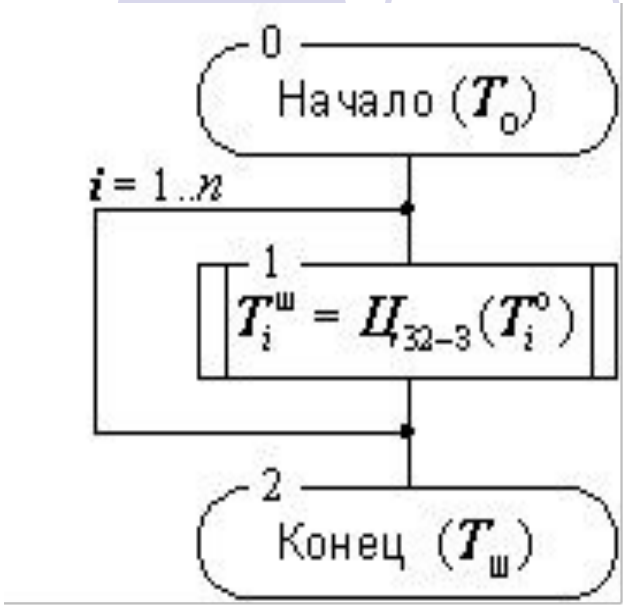
Основные режимы шифрования.

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

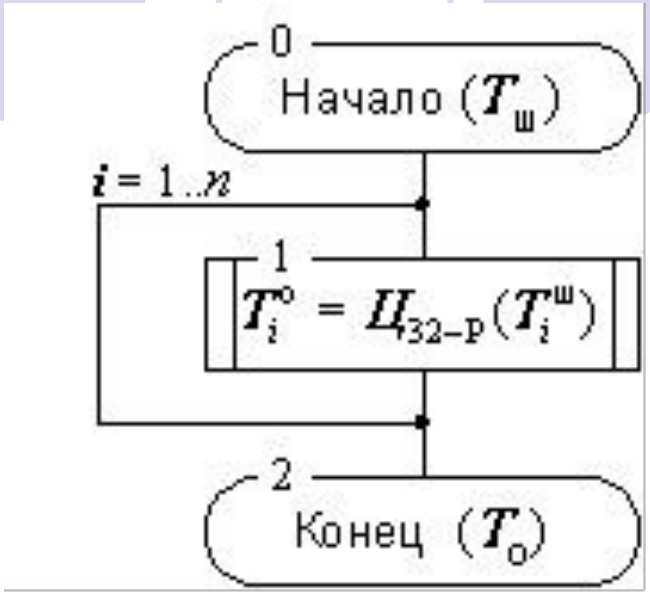
- простая замена,
- гаммирование,
- гаммирование с обратной связью,
- дополнительный режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита. Однако в двух режимах гаммирования есть возможность обработки неполного блока данных размером меньше 8 байт, что существенно при шифровании массивов данных с произвольным размером, который может быть не кратным 8 байтам.

Простая замена.



Алгоритм зашифрования данных в режиме простой замены



Алгоритм расшифрования данных в режиме простой замены

Особенности:

1. При зашифровании одинаковых блоков открытого текста получаются одинаковые блоки шифртекста и наоборот
2. Если длина шифруемого массива данных не кратна 8 байтам или 64 битам, возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит.

Гаммирование.

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, для получения зашифрованных (открытых) данных. В ГОСТе для этой цели используется операция побитового сложения по модулю 2.

Гаммирование решает проблемы:

Во-первых, все элементы гаммы различны для реальных шифруемых массивов и, следовательно, результат зашифрования даже двух одинаковых блоков в одном массиве данных будет различным.

Во-вторых, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока.

РГПЧ, используемый для выработки гаммы, является рекуррентной функцией:

$\Omega_{i+1}=f(\Omega_i)$, где Ω_i – элементы рекуррентной последовательности, *f* – функция преобразования.

Элемент Ω_0. является параметром алгоритма для режимов гаммирования, на схемах он обозначен *S*, и называется в криптографии ***синхропосылкой*** , а в ГОСТе – ***начальным заполнением*** одного из регистров шифрователя. По определенным соображениям разработчики ГОСТа решили использовать для инициализации РГПЧ не непосредственно синхропосылку, а результат ее преобразования по циклу 32-3:

$$\Omega_0=U_{32\text{-}3}(S).$$

Элементы последовательности, вырабатываемой РГПЧ, являются функцией своего номера и начального заполнения РГПЧ: $\Omega_i = f_i(\Omega_0)$, где $f_i(X) = f(f_{i-1}(X))$, $f_0(X) = X$. С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа:

$$\Gamma_i = U_{32\text{-}3}(\Omega_i) = U_{32\text{-}3}(f_i(\Omega_0)) = U_{32\text{-}3}(f_i(U_{32\text{-}3}(S))) = \varphi_i(S,K),$$

где Γ_i – i-тый элемент гаммы, K – ключ.

Последовательность элементов гаммы однозначно определяется ключом и синхропосылкой. Для обратимости процедуры шифрования в процессах за- и расшифрования должна использоваться одна и та же синхропосылка. Это приводит к необходимости хранить или передавать синхропосылку по каналам связи вместе с зашифрованными данными, хотя она может быть предопределена или вычисляться особым образом, если исключается шифрование двух массивов на одном ключе.

- РГПЧ спроектирован разработчиками ГОСТа исходя из необходимости выполнения следующих условий:
- период повторения последовательности чисел, вырабатываемой РГПЧ, не должен сильно (в процентном отношении) отличаться от максимально возможного при заданном размере блока значения 2^{64};
- соседние значения, вырабатываемые РГПЧ, должны отличаться друг от друга в каждом байте, иначе задача криптоаналитика будет упрощена;
- РГПЧ должен быть достаточно просто реализуем как аппаратно, так и программно на наиболее распространенных типах процессоров, большинство из которых, как известно, имеют разрядность 32 бита.

Характеристики РГПЧ

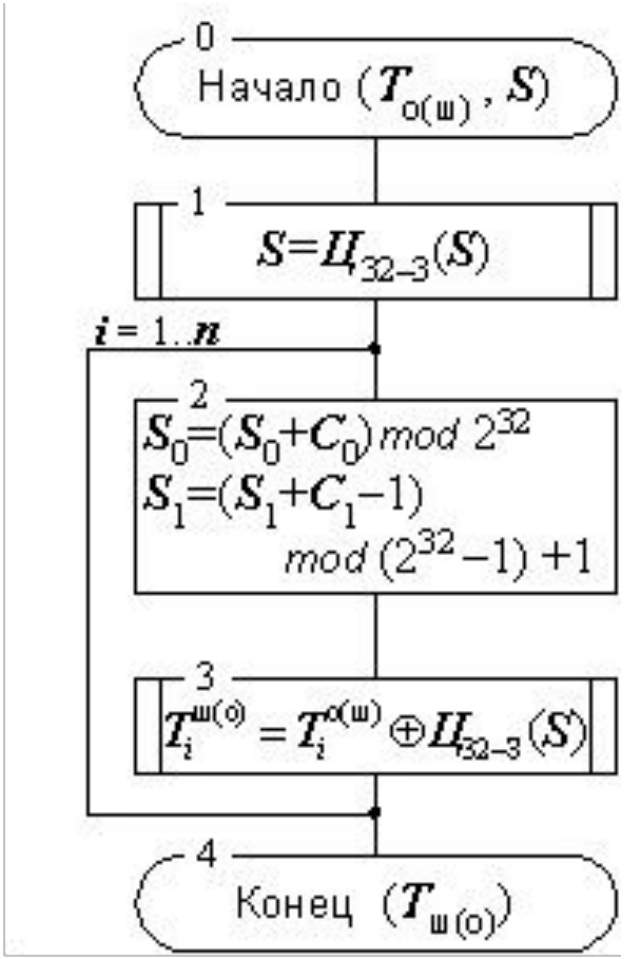
- в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга:

$$\Omega_i = (\Omega_i^0, \Omega_i^1),\ |\Omega_i^0| = |\Omega_i^1| = 32,\ \Omega_{i+1}^0 = \hat{f}(\Omega_i^0),\ \Omega_{i+1}^1 = \hat{f}(\Omega_i^1)$$

фактически, существуют два независимых РГПЧ для старшей и младшей частей блока.

- рекуррентные соотношения для старшей и младшей частей следующие:
- $\Omega_{i+1}^0 = (\Omega_i^0 + C_0) mod 2^{32}$, где $C_0 = 1010101_{16}$;
- $\Omega_{i+1}^1 = (\Omega_i^1 + C_1 - 1) mod (2^{32} - 1) + 1$, где $C_1 = 1010104_{16}$;
- период повторения последовательности для младшей части составляет 2^{32}, для старшей части $2^{32}–1$, для всей последовательности период составляет $2^{32}(2^{32}–1)$,

Алгоритм зашифрования (расшифрования) данных в режиме гаммирования



$T_{о(ш)}$ – массив открытых (зашифрованных) данных произвольного размера, подвергаемый процедуре зашифрования (расшифрования), по ходу процедуры массив подвергается преобразованию порциями по 64 бита;

S* – *синхропосылка , 64-битовый элемент данных, необходимый для инициализации генератора гаммы;

Шаг 1

Начальное преобразование синхропосылки, выполняемое для ее «рандомизации», результат используется как начальное заполнение РГПЧ;

Шаг 2

Один шаг работы РГПЧ, реализующий его рекуррентный алгоритм. В ходе данного шага старшая (S_1) и младшая (S_0) части последовательности данных вырабатываются независимо друг от друга;

Шаг 3

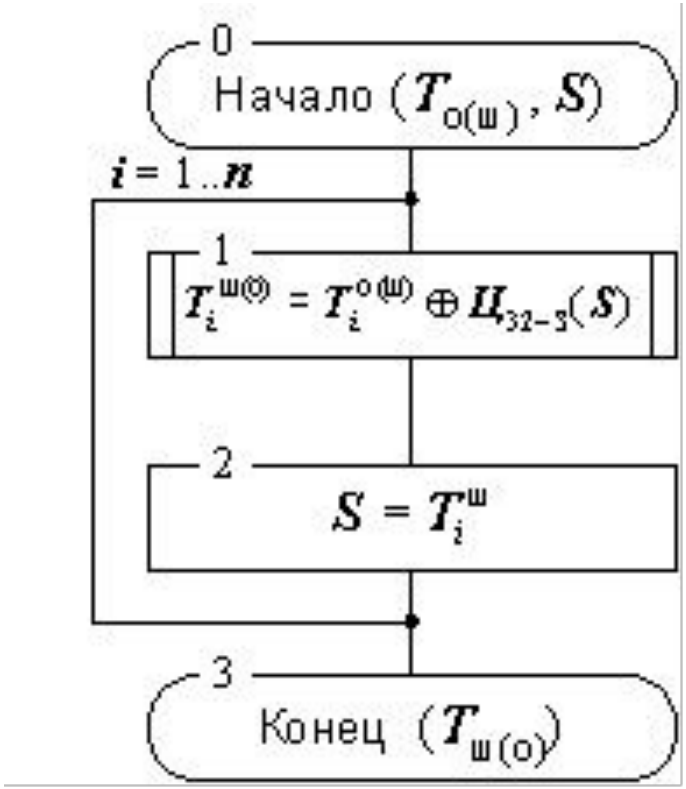
Гаммирование. Очередной 64-битовый элемент, выработанный РГПЧ, подвергается процедуре

ОСОБЕННОСТИ ГАММИРОВАНИЯ КАК РЕЖИМА ШИФРОВАНИЯ:

- Одинаковые блоки в открытом массиве данных дадут при зашифровании различные блоки шифртекста.
- Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо Так, для зашифрования неполного блока в 1 бит согласно стандарту следует использовать самый младший бит из блока гаммы.
- Синхропосылка, использованная при зашифровании, должна быть передана для использования при расшифровании. Это может быть достигнуто следующими путями:
- хранить или передавать синхропосылку вместе с зашифрованным массивом данных, что приведет к увеличению размера массива данных при зашифровании на размер синхропосылки, то есть на 8 байт;
- использовать предопределенное значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует

Гаммирование с обратной связью

Данный режим иногда называется ***гаммированием с зацеплением блоков*** . На стойкость шифра факт зацепления блоков не оказывает никакого влияния.



Выработка имитовставки к массиву данных.

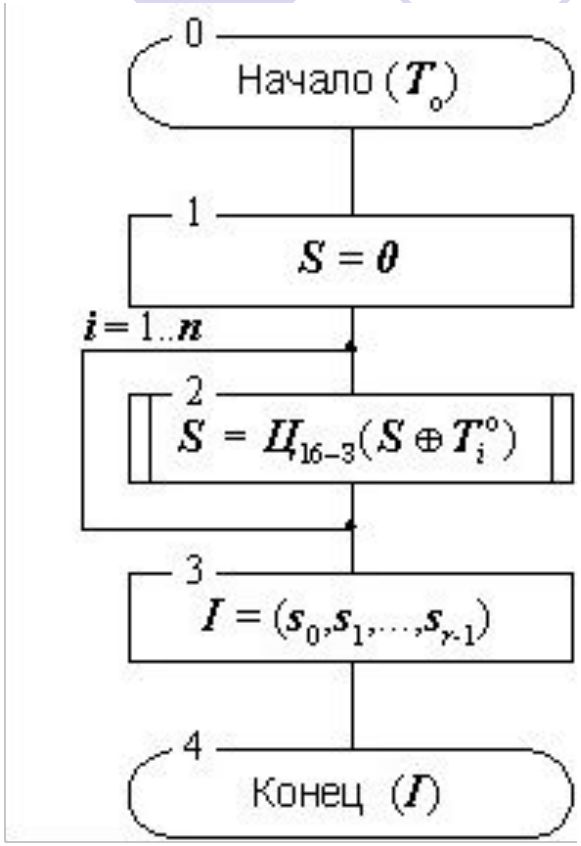
Для решения задачи обнаружения искажений в зашифрованном массиве данных с заданной вероятностью в ГОСТе предусмотрен дополнительный режим криптографического преобразования – выработка имитовставки.

- Имитовставка – это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации.
- Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации.

Для потенциального злоумышленника две следующие задачи практически неразрешимы, если он не владеет ключом:

- вычисление имитовставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитовставку;

Схема алгоритма выработки имитовставки



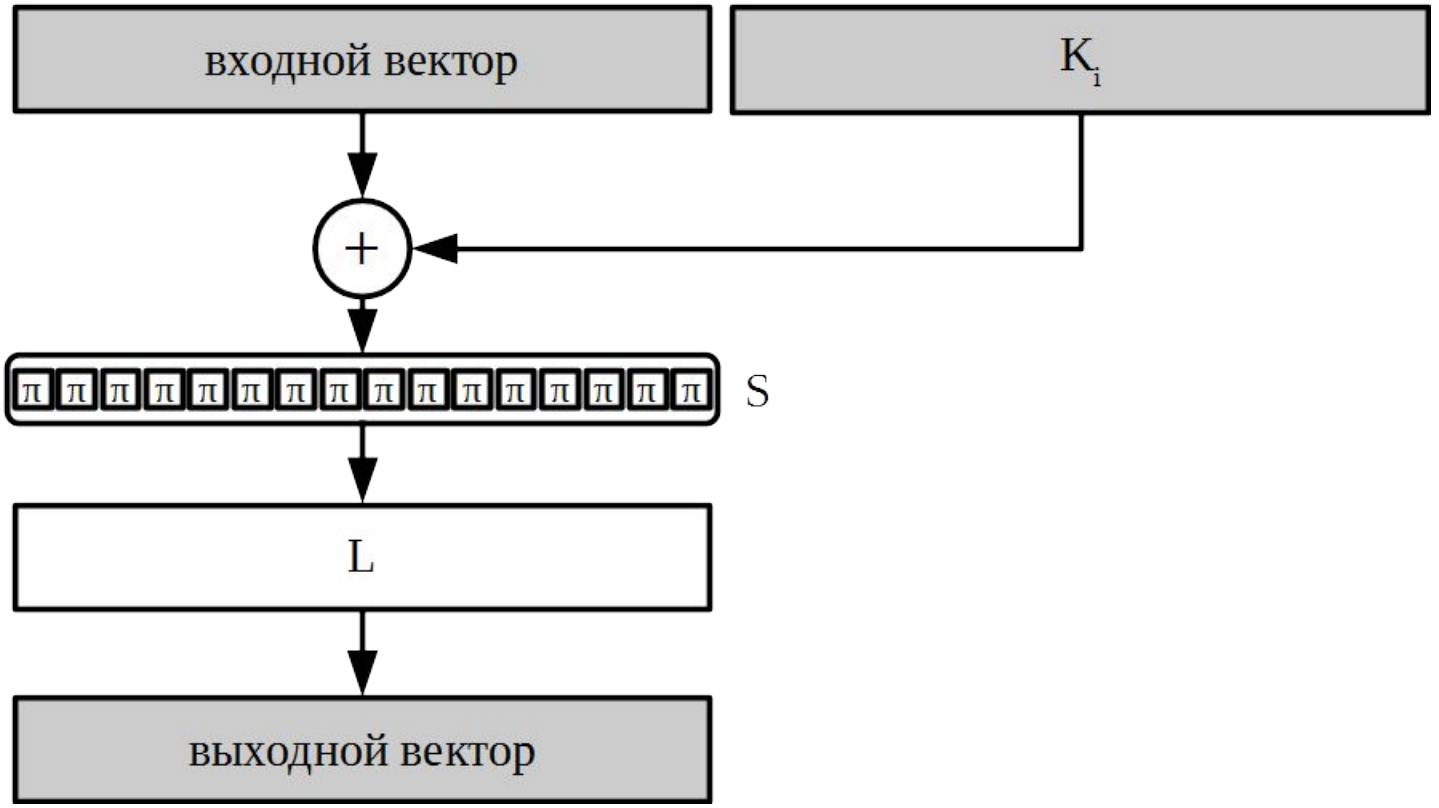
В качестве имитовставки берется часть блока, полученного на выходе, обычно – 32 его младших бита. При выборе размера имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных равна величине $2^{-|I|}$ на одну попытку подбора, если в распоряжении злоумышленника нет более эффективного метода подбора, чем простое угадывание. При использовании имитовставки размером 32 бита эта вероятность равна $2^{-32} \approx 0.23 \cdot 10^{-9}$.

- В режиме обычного гаммирования изменения в определенных битах шифртекста влияют только на соответствующие биты открытого текста
- В режиме гаммирования с обратной связью блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифрованный блок, то после расшифрования искаженными окажутся два блока открытых данных – соответствующий и следующий за ним, в соответствующем блоке открытых данных искаженными окажутся те же самые биты, что и в блоке шифрованных данных, а в следующем блоке открытых данных все биты независимо друг от друга с вероятностью 1⁄2 изменят свои значения.

«Кузнечик»

- В отличие от ГОСТ 28147-89 новый шифр представляет собой не сеть Фейстеля, а т.н. SP-сеть: преобразование, состоящее из нескольких одинаковых раундов, при этом каждый раунд состоит из нелинейного и линейного преобразований, а также операции наложения ключа. В отличие от сети Фейстеля, при использовании SP-сети преобразуется весь входной блок, а не его половина.
- Длина входного блока «Кузнечика» — 128 бит, ключа — 256 бит. Новый шифр существенно отличается от старого, среди его основных достоинств можно выделить
- вдвое увеличенную длину блока (*128 бит, или 16 байт*, против 64 бит, или 8 байт),
- нетривиальное ключевое расписание (*сеть Фейстеля как ключевое расписание* против использования частей секретного ключа в качестве цикловых ключей),
- сокращенное число циклов (*10 циклов* против 32 циклов),
- принципиально иное устройство самого шифра (*LSX-шифр* против сети Фейстеля).

Шифр принадлежит к классу LSX-шифров: его базовое преобразование (функция шифрования блока) представляется десятью циклами последовательных преобразований
L (линейное преобразование), *S* (подстановка) и *X* (смешивание с цикловым ключом):
Раундовое преобразование можно изобразить следующим образом:



Преобразования

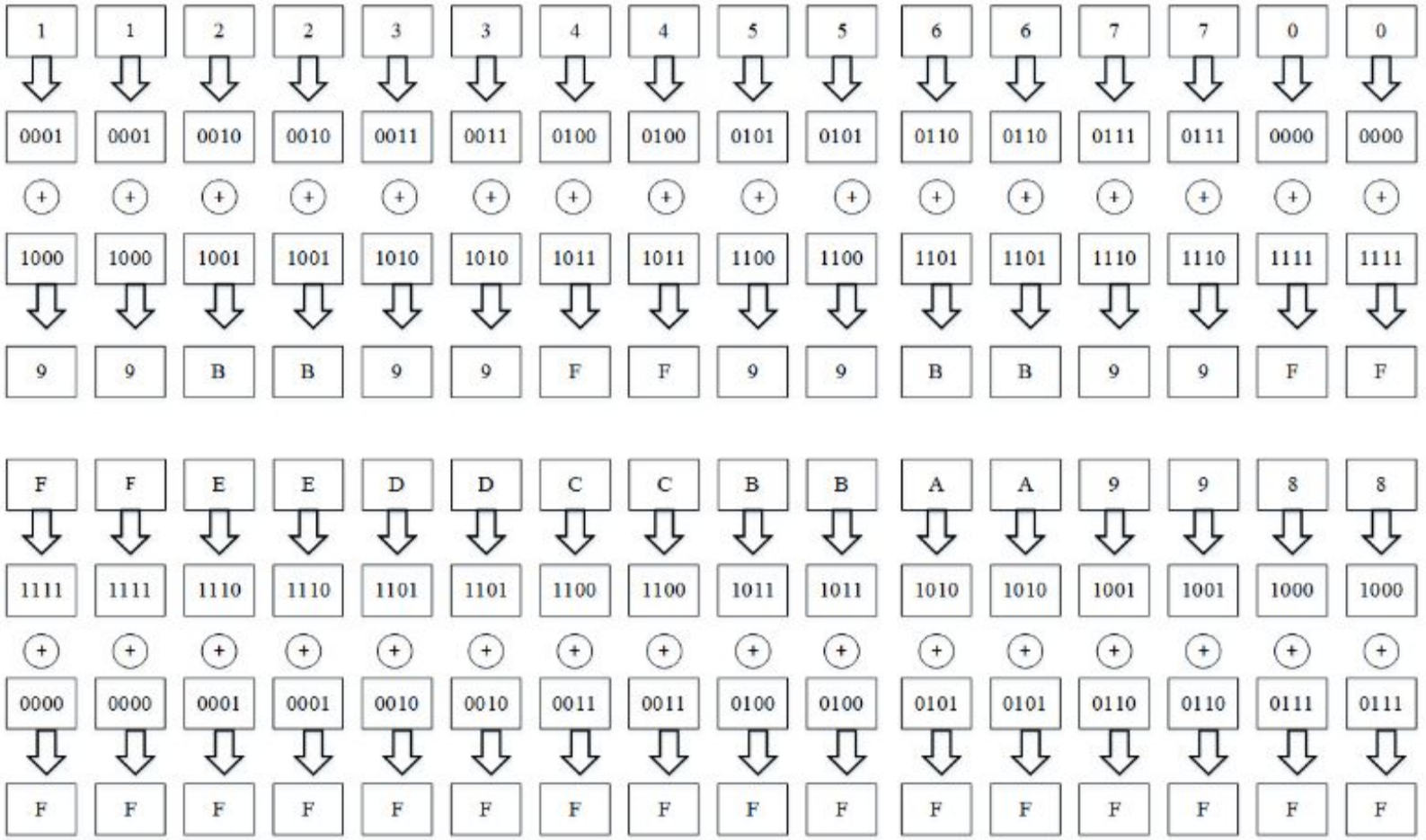
- Шифрование основано на последовательном применении нескольких однотипных раундов, каждый из которых содержит три преобразования: сложение с раундовым ключом, преобразование блоком подстановок и линейное преобразование.

- Первое преобразование - 128-битный входной вектор очередного раунда складывается побитно с раундовым ключом:

$X[k]: V_{128} \to V_{128}$ $X[k](a) = k \oplus a$, где $k, a \in V_{128}$;

- За счет преобразования всего блока данных на каждом шаге обеспечивается гораздо более быстрое перемешивание входных данных по сравнению со схемой Фейстеля. Кроме того, слой линейного преобразования конструируется таким образом, чтобы обеспечить максимально возможное рассеивание

Имеется открытый текст *a* = ***1122334455667700ffeeddccbbaa9988*** и мастер-ключ *key* = ***8899aabbccddeeff0011223344556677fedcba98765432100123456789abcdef***.
Первое преобразование — это побитное сложение по модулю 2 открытого текста и первого раундового ключа,



Преобразование X: a-открытый текст, b-раундовый ключ (совпадает со старшей частью мастер-ключа), outdata-результат преобразования.

Результатом первого преобразования является вектор длиной 128 бит он равен **99BB99FF99BB99FFFFFFFFFFFFFFFF**.

- Второе преобразование алгоритма — это нелинейное биективное преобразование вектора полученного после первой операции с использованием блока подстановок.
- 128-битовый блок *a* алгоритма шифрования представляется в виде последовательности 16 байтов, которые нумеруются справа налево, начиная с 0:

$$S: V_{128} \to V_{128}$$

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

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

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

Поскольку каждый байт a_t может принимать значение от 0 до 255, массив π' имеет 256 элементов. Это значит, что при a_t = 0 значение a_j будет заменено на 5(0) = 252, при a_t = 1 — на 5(1) = 238 и т.д. (все числа записаны в десятеричном виде).

- Линейное преобразование может быть реализовано не только как обычно в блочных шифрах — матрицей, но и с помощью РСЛОС — линейного регистра сдвига с обратной связью, который движется 16 раз.

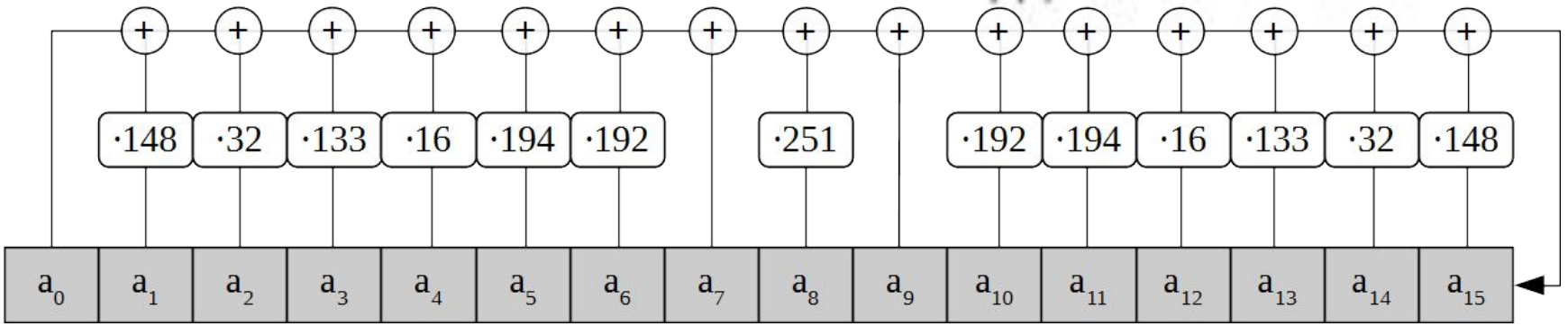
$R: V_{128} \to V_{128}$ $\quad R(a) = R(a_{15}||\ldots||a_0) = \ell(a_{15}, \ldots, a_0)||a_{15}||\ldots||a_1,$

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

$L: V_{128} \to V_{128}$ $\quad L(a) = R^{16}(a)$, где $a \in V_{128}$;

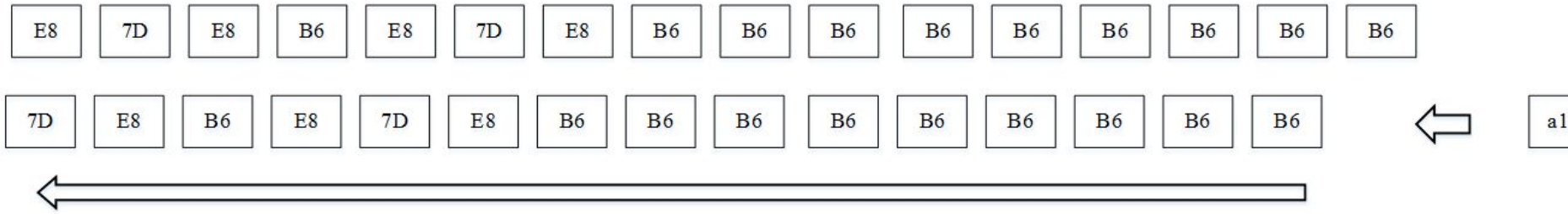
- Сам регистр реализуется над полем Галуа по модулю неприводимого многочлена степени 8:

$$p(x) = x^8 + x^7 + x^6 + x + 1$$



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

- Линейное преобразование, выполняемое с использованием линейного регистра сдвига с обратной связью : сначала результат S-преобразования побайтно считывается, затем каждый считанный байт умножается на 256 (это необходимо для вычисления позиции числа в таблице table.h со всеми возможными результатами умножения в поле GF(2^n) согласно ГОСТа). Из позиции считывается число, к нему прибавляется коэффициент **148, 32, 133, 16, 194, 192, 1, 251, 1, 192, 194, 16, 133, 32, 148, 1** в зависимости от номера итерации, так происходит со следующими байтами. Байты складываются между собой по модулю два и все 128 бит (результат S-преобразования) сдвигаются в сторону младшего разряда, а полученное число в шестнадцатеричном виде записывается на место считанного байта. Регистр сдвигается и перезаписывается 16 раз.



Результатом L-преобразования будет следующий 128-битный вектор **E297B686E355B0A1CF4A2F9249140830** Это результат работы первого раунда алгоритма, таким же образом будут проходить последующие 8 раундов. Заключительный 10 раунд включает в себя только X-преобразование результатов работы 9 раундов и ключа 10 раунда.

Выработка раундовых ключей

- Первые два получаются разбиением мастер-ключа пополам. Далее для выработки очередной пары раундовых ключей используется 8 итераций сети Фейстеля, где, в свою очередь, в качестве раундовых ключей используется счетчиковая последовательность, прошедшая через линейное преобразование алгоритма:

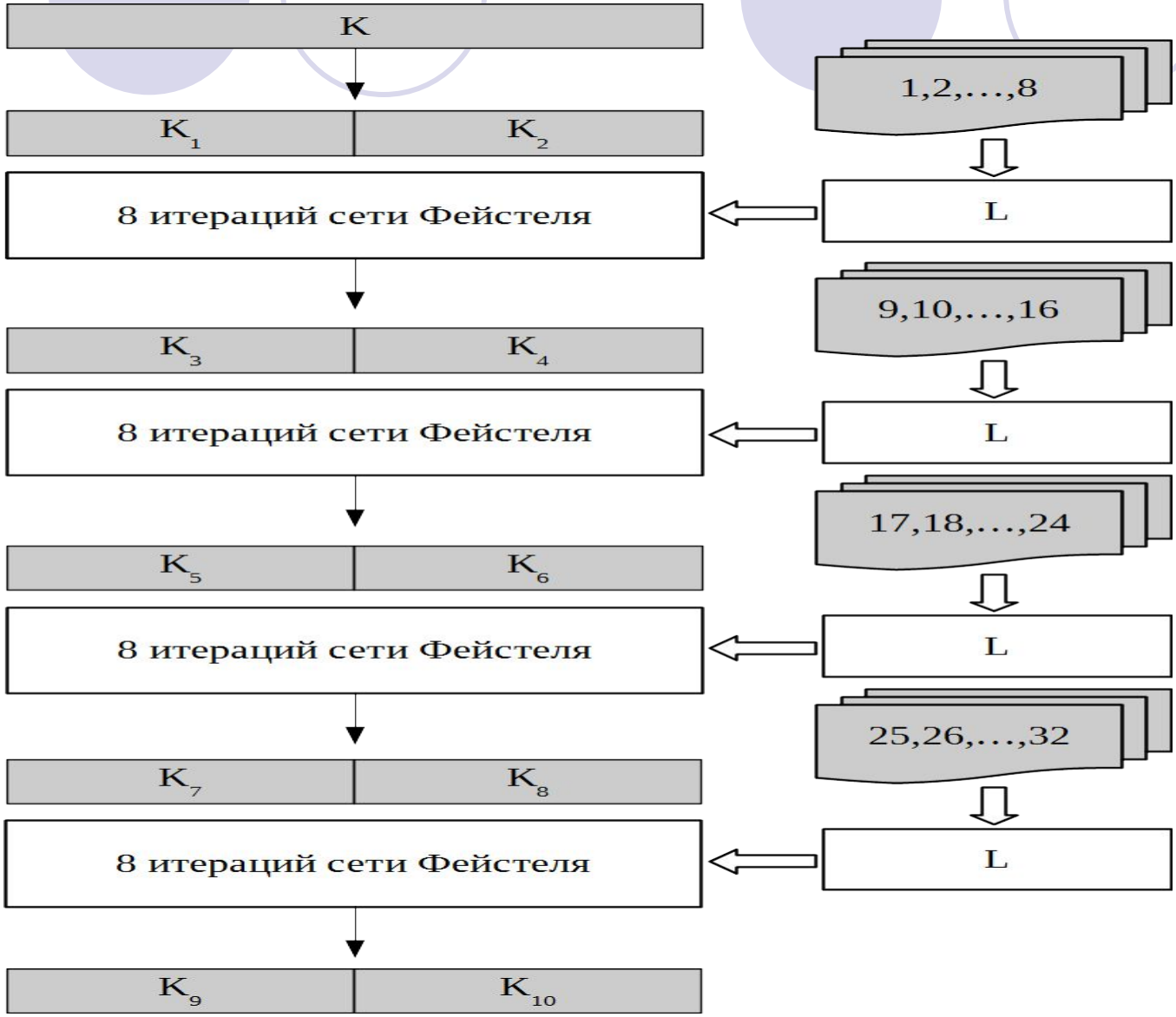
$F[k]: V_{128} \times V_{128} \rightarrow V_{128} \times V_{128}$

$F[k](a_1, a_0) = (LSX[k](a_1) \oplus a_0, a_1)$,
где $k, a_0, a_1 \in V_{128}$.

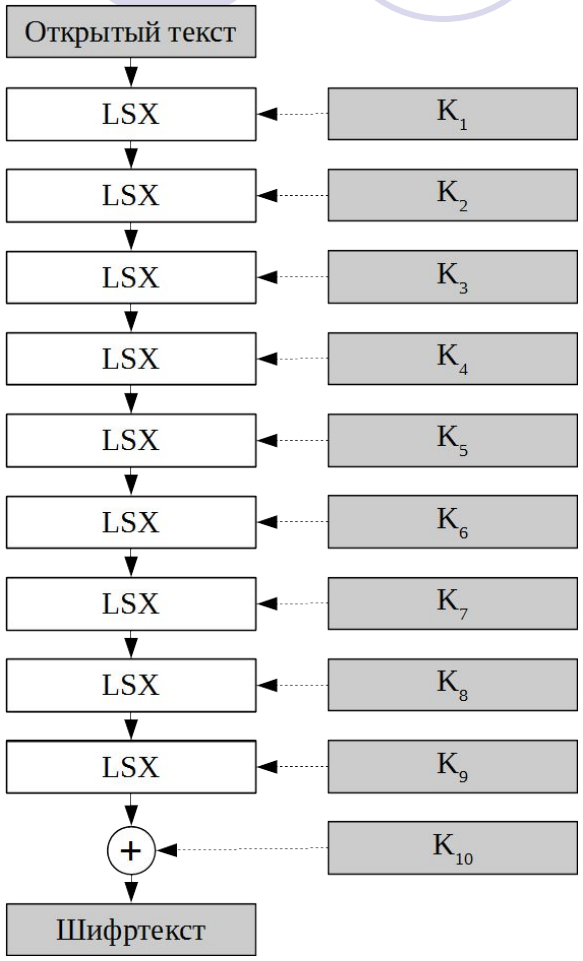
$$C_i = L(\mathrm{Vec}_{128}(i)),\ i = 1, 2, \ldots, 32.$$

$$(K_{2i+1}, K_{2i+2}) = F[C_{8(i-1)+8}] \ldots F[C_{8(i-1)+1}](K_{2i-1}, K_{2i}),\ i = 1, 2, 3, 4.$$

- процедура выработки раундовых ключей



Шифрование и расшифрование



шифрование одного 128-битного входного блока описывается следующим уравнением:

$$E_{K_1,\ldots,K_{10}}(a) = X[K_{10}]LSX[K_9]\ldots LSX[K_2]LSX[K_1](a),$$

Расшифрование реализуется обращением базовых преобразований и применением их в обратном порядке:

$$D_{K_1,\ldots,K_{10}}(a) = X[K_1]S^{-1}L^{-1}X[K_2]\ldots S^{-1}L^{-1}X[K_9]S^{-1}L^{-1}X[K_{10}](a),$$

Криптоанализ блочных шифров

Как и все шифры, алгоритмы которых известны, блочные шифры подвергаются криптографическим атакам. Цель атаки — разработать алгоритм взлома более эффективный, чем полный перебор всех возможных ключей. В случае нахождения подобного решения, атака считается успешной. При этом шифр является взломанным, если существуют атака, позволяющая провести взлом за время, в течение которого информация сохраняет актуальность, и проведение подобной атаки выгодно злоумышленнику.

Криптоанализ ставит своей задачей в разных условиях получить дополнительные сведения о ключе шифрования, чтобы значительно уменьшить диапазон вероятных ключей. Результаты криптоанализа могут варьироваться по степени практической применимости. Так, криптограф Ларс Кнудсен [36] предлагает следующую классификацию успешных исходов криптоанализа блочных шифров в зависимости от объема и качества секретной информации, которую удалось получить:

- Полный взлом – криптоаналитик извлекает секретный ключ.
- Глобальная дедукция – криптоаналитик разрабатывает функциональный эквивалент исследуемого алгоритма, позволяющий зашифровывать и расшифровывать информацию без знания ключа.
- Частичная дедукция – криптоаналитику удается расшифровать или зашифровать некоторые сообщения.
- Информационная дедукция – криптоаналитик получает некоторую информацию об открытом тексте или ключе.

- **Атака на основе только шифртекста.** Криптоаналитик располагает шифртекстами $y_1, \ldots, y_m$, полученными из неизвестных открытых текстов $x_1, \ldots, x_m$ различных сообщений. Требуется найти хотя бы один из x_i, $i = \overline{1, m}$ (или соответствующий ключ k_i), исходя из достаточного числа m криптограмм, или убедиться в своей неспособности сделать это. В качестве частных случаев возможно совпадение ключей: $k_1 = \cdots = k_m$ или совпадение открытых текстов: $x_1 = \cdots = x_m$.
- **Атака на основе открытого текста.** Криптоаналитик располагает парами $(x_1, y_1), \ldots, (x_m, y_m)$ открытых и соответствующим им зашифрованных текстов. Требуется определить ключ k_i для хотя бы одной из пар. В частном случае, когда $k_1 = \cdots = k_m = k$, требуется определить ключ k или, убедившись в своей неспособности сделать это, определить открытый текст x_{m+1} еще одной криптограммы y_{m+1}, зашифрованной на том же ключе.
- **Атака на основе подобранного открытого текста** отличается от предыдущей лишь тем, что криптоаналитик имеет возможность выбора открытых текстов $x_1, \ldots, x_m$. Цель атаки та же, что и предыдущей. Подобная атака возможна, например, в случае, когда криптоаналитик имеет доступ к шифратору передающей стороны.
- **Атака на основе адаптивно подобранного открытого текста.** Это частный случай вышеописанной атаки с использованием подобранного открытого текста. Криптоаналитик может не только выбирать используемых шифруемый текст, но также уточнять свой последующий выбор на основе полученных ранее результатов шифрования.

Полный перебор

- **Полный перебор** (или **метод «грубой силы»**) — метод решения математических задач. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий.
- В криптографии на вычислительной сложности полного перебора основывается оценка криптостойкости шифров. В частности, шифр считается криптостойким, если не существует метода «взлома» существенно более быстрого чем полный перебор всех ключей. Криптографические атаки, основанные на методе полного перебора, являются самыми универсальными, но и самыми долгими.

Дифференциальный криптоанализ

Подобной атаке подвержены шифры с постоянным S-блоком и шифрование в режиме кодовой электронной книги. Данный метод работает с парами шифротекстов, для которых известно различие соответствующих открытых текстов, и рассматривает эволюцию этих различий.

Дифференциальный криптоанализ основан на изучении преобразования разностей между шифруемыми значениями на различных раундах шифрованияДифференциальный криптоанализ основан на изучении преобразования разностей между шифруемыми значениями на различных раундах шифрования. В качестве разности, как правило, применяется операция побитового суммирования по модулю 2Дифференциальный криптоанализ основан на изучении преобразования разностей между шифруемыми значениями на различных раундах шифрования. В качестве разности, как правило, применяется операция побитового суммирования по модулю 2, хотя существуют атаки и с вычислением разности по модулюДифференциальный криптоанализ основан на изучении преобразования разностей между шифруемыми значениями на различных раундах шифрования. В качестве разности, как правило, применяется операция побитового суммирования по модулю 2, хотя существуют атаки и с вычислением разности по модулю . Является статистической атакой. Применим для взлома DESДифференциальный

Линейный криптоанализ

- Криптоанализ происходит в два шага. Первый — построение соотношений между открытым текстом, шифртекстом и ключом, которые справедливы с высокой вероятностью. Второй — использование этих соотношений вместе с известными парами открытый текст — шифртекст для получения битов ключа.

Интегральный криптоанализ

- Интегральный криптоанализ — вид атак, особенно применимый к блочный шифрам, построенным на SP-сети. В отличие от дифференциального криптоанализа, использующего пару выбранного открытого текста с фиксированным разницей, вычисленной при помощи операции XOR, интегральный криптоанализ использует множества открытых текстов, в которых одни части удерживаются постоянными, в то время как другие варьируются среди всевозможных значений. Подобное множество с необходимостью имеет сумму по модулю 2 (XOR), равной 0, в то время, как соответствующая сумма шифротекста содержит информацию об операциях шифра.