

МЕТОДЫ СИММЕТРИЧНОГО ШИФРОВАНИЯ

Алгоритм усовершенствованного стандарта шифрования
AES

Конкурс AES

- Предпосылки:
 - Недостаточная длина ключа DES (фактически 56 бит)
 - Ориентация DES на аппаратную реализацию
 - Успешные криптоатаки на DES
- Конкурс на новый стандарт (*Advanced Encryption Standard*) объявлен 12 сентября 1997 года институтом NIST (*The National Institute of Standards and Technology, USA*):
 - Заявки принимались от любой организация или группы исследователей
 - Шифр-победитель должен распространяться по всему миру на не эксклюзивных условиях и без платы за пользование патентом
 - Минимальные требования к новому стандарту касались типа алгоритм, длины блока обработки и допустимых размеров ключей

Требования к новому алгоритму

- Алгоритм должен быть симметричным
- Алгоритм должен быть блочным шифром
- Алгоритм должен иметь длину блока 128 бит, и поддерживать три длины ключа : 128, 192 и 256 бит.
- Дополнительные требования:
 - Использовать операции, легко реализуемые, как аппаратно (на микрочипах), так и программно
 - Ориентироваться на 32-разрядные процессоры
 - Не усложнять без необходимости структуру шифра для того, чтобы все заинтересованные стороны были в состоянии убедиться, что в нём не заложено каких-либо недокументированных возможностей

Этапы конкурса

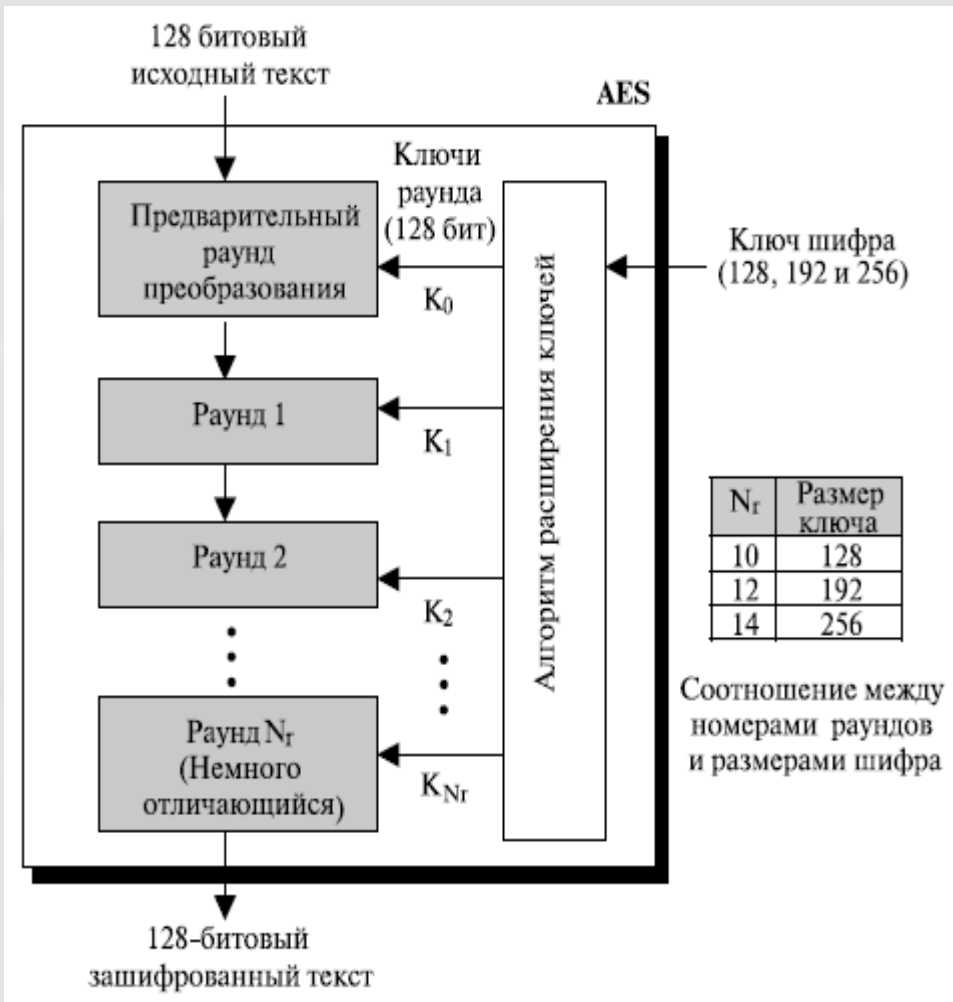
- Предварительный отбор: 20.08.1998 года на 1-й конференции AES был объявлен список из 15 кандидатов: *CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, Twofish*
- Первый раунд (128 битные ключи):
 - Исследовались криптографическая стойкость, а также практические аспекты реализации: оптимизацию скорости выполнения и размера кода на различных архитектурах (от ПК до смарт-карт и аппаратных реализаций)
 - В августе 1999 года были объявлены 5 финалистов: *MARS, RC6, Rijndael, Serpent u Twofish*
- Второй раунд (192-битные и 256-битные ключи):
 - Оценивалась пригодность финалистов для использования в качестве генераторов случайных чисел. Продолжительность статистических тестов NIST составила несколько месяцев
 - По результатам полученных на обоих раундах характеристик шифров проведено голосование

Итоги конкурса AES

| | Rijndael | Serpent | TwoFish | RC-6 | MARS |
|---|----------|---------|---------|---------|---------|
| Аппаратная реализации (эффективность) | + | + | | | |
| Программная реализация (относительное быстродействие) | 3.5 | 1 | 5.5 | 6 | 4 |
| Этап расширения ключа | + | | + | | |
| Возможность распараллеливания | + | | | | |
| Запас криптостойкости | Оптимум | Завышен | Завышен | Оптимум | Оптимум |
| Кол-во голосов: ЗА/ПРОТИВ | 86/10 | 59/7 | 31/21 | 23/37 | 13/83 |

Шифр *RIJNDAEL* официально объявлен победителем конкурса в день 02.10.2000 и получил второе наименование *ADVANCED ENCRYPTION STANDARD*

Шифр AES

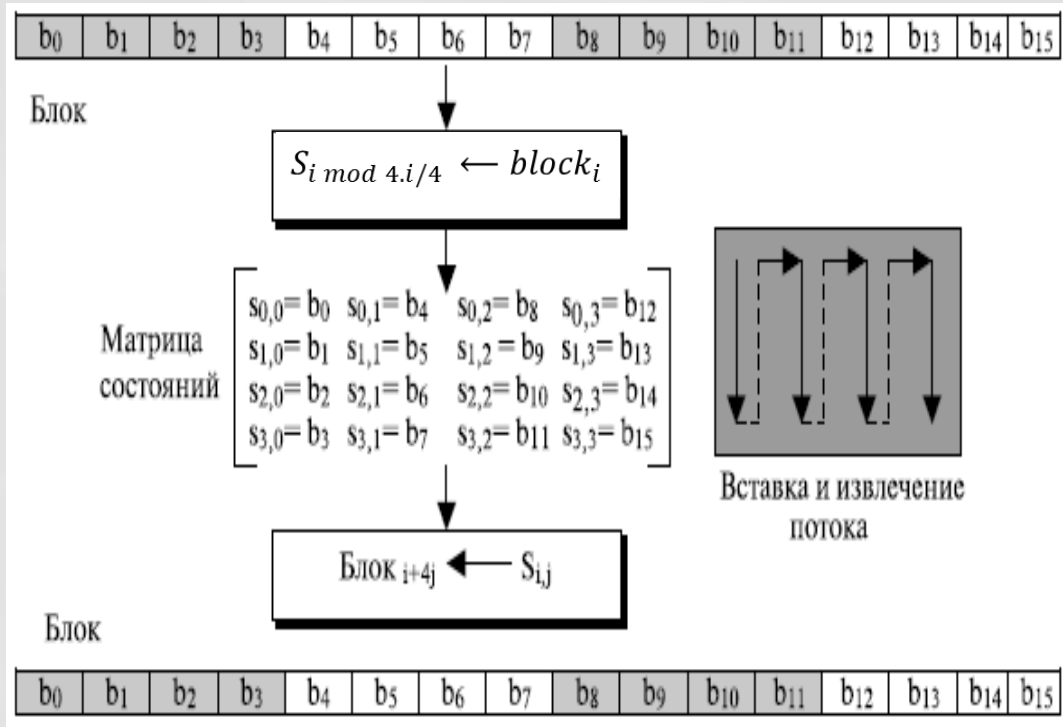


- Используется не сеть Фейстеля, а *Square – like* структуру(структуру «квадрат»)
- Поддерживаются три версии (с 10, 12 и 14 раундами). Каждая версия использует различный размер ключа шифра (128, 192 или 256) и имеет соответствующее название: AES-128, AES-196, AES-256
- Ключи раунда имеют тот же самый размер, что и блоки зашифрованного или исходного текста - всегда 128 бит

Единицы данных

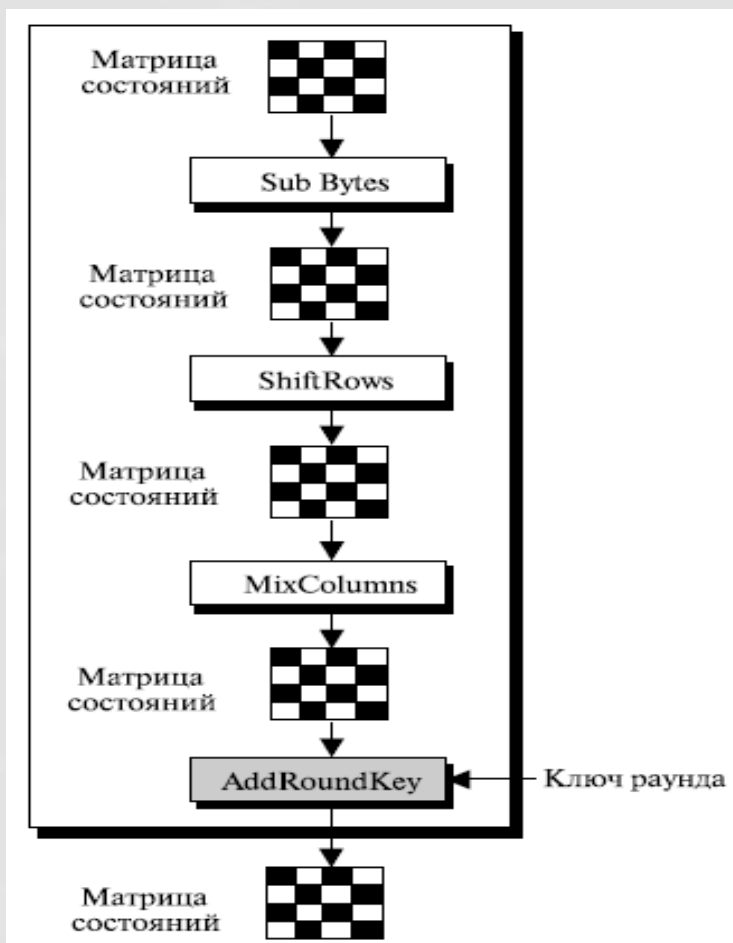
- Байт — последовательность из 8 битов . Операции над байтами производятся, как над элементами поля Галуа $GF(2^8)$, то есть байту $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ соответствует полином $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$
 - Операция сложения полиномов выполняется, как XOR коэффициентов при одинаковых степенях (XOR байтов)
 - Операция умножения полиномов выполняется по модулю неприводимого полинома $x^8 + x^4 + x^3 + x + 1$ (XOR с $11B_{16}$)
- Блок — последовательность из 16 байтов, над которой оперирует алгоритм
- Ключ — последовательность из 16, 24 или 32 байтов

Матрица состояний



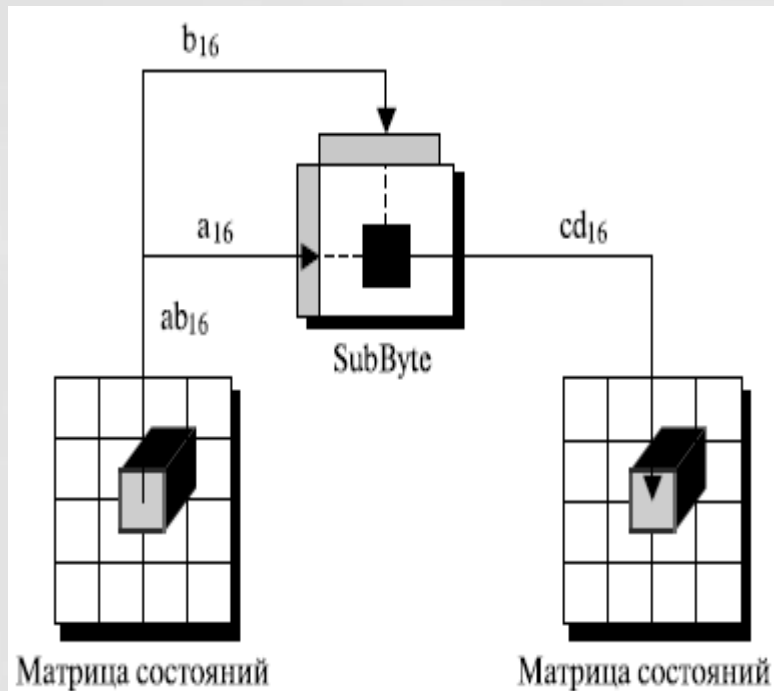
- Размер 4x4 байт
- Содержит промежуточные результаты обработки (состояния) блоков данных

Раунд AES



- Раунд состоит из четырех различных обратимых преобразований, называемых слоями.
- Каждое преобразование модифицирует матрицу состояний
- Каждый слой разрабатывался с учетом противодействия линейному и дифференциальному криптоанализу:
 - *SubBytes* - слой подстановок (замен) байтов для обеспечения нелинейных свойств шифра
 - *ShiftRows* - слой линейного перемешивания строк
 - *MixColumns* - слой линейного перемешивания столбцов
 - *AddRoundKey* - слой рандомизации с выполнением XOR промежуточного состояния с ключом раунда.

Преобразование «Подстановка»



- Процедура *SubByte* выполняет 16 независимых преобразований байта в байт (если два байта имеют одинаковое значение, то они преобразуются одинаково)
- Содержание каждого байта изменяется, но расположение байтов в матрице остается тем же самым
- Допустимые реализации:
 - Таблицей подстановки (S-блок) 16x16
 - Математическим вычислением -заменой байта аффинным преобразованием его мультипликативной инверсии в $GF(2^8)$ по модулю неприводимого многочлена $x^8+x^4+x^3+x+1$
- В процессе расшифровки используется процедура *InvSubByte* – инверсная по отношению *SubByte*

Таблица подстановок S-box

$$b_{16} = 3_{16}$$



Прямая подстановка

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 3 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 4 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 5 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

$$a_{16} = b_{16}$$



$$cd_{16} = 6D_{16}$$



Обратная подстановка

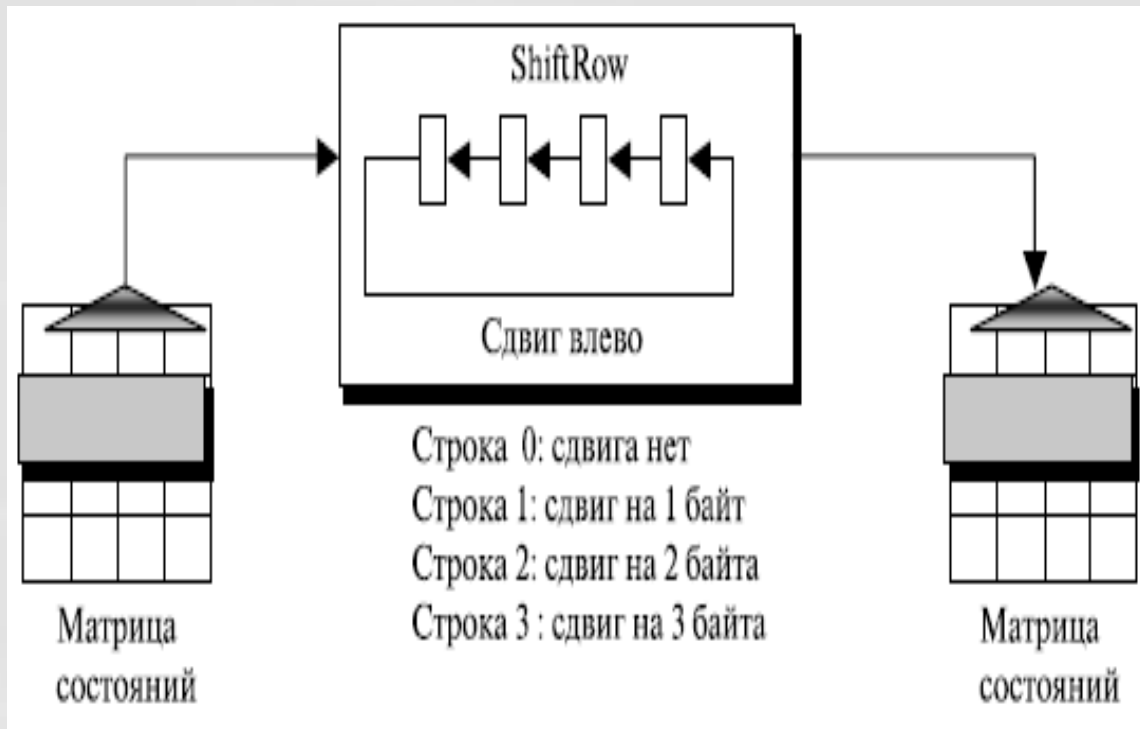
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | 52 | 09 | 6a | d5 | 30 | 36 | a5 | 38 | bf | 40 | a3 | 9e | 81 | f3 | d7 | fb |
| 1 | 7c | e3 | 39 | 82 | 9b | 2f | ff | 87 | 34 | 8e | 43 | 44 | c4 | de | e9 | cb |
| 2 | 54 | 7b | 94 | 32 | a6 | c2 | 23 | 3d | ee | 4c | 95 | 0b | 42 | fa | c3 | 4e |
| 3 | 08 | 2e | a1 | 66 | 28 | d9 | 24 | b2 | 76 | 5b | a2 | 49 | 6d | 8b | d1 | 25 |
| 4 | 72 | f8 | f6 | 64 | 86 | 68 | 98 | 16 | d4 | a4 | 5c | cc | 5d | 65 | b6 | 92 |
| 5 | 6c | 70 | 48 | 50 | fd | ed | b9 | da | 5e | 15 | 46 | 57 | a7 | 8d | 9d | 84 |
| 6 | 90 | d8 | ab | 00 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 05 | b8 | b3 | 45 | 06 |
| 7 | d0 | 2c | 1e | 8f | ca | 3f | 0f | 02 | c1 | af | bd | 03 | 01 | 13 | 8a | 6b |
| 8 | 3a | 91 | 11 | 41 | 4f | 67 | dc | ea | 97 | f2 | cf | ce | f0 | b4 | e6 | 73 |
| 9 | 96 | ac | 74 | 22 | e7 | ad | 35 | 85 | e2 | f9 | 37 | e8 | 1c | 75 | df | 6e |
| a | 47 | f1 | 1a | 71 | 1d | 29 | c5 | 89 | 6f | b7 | 62 | 0e | aa | 18 | be | 1b |
| b | fc | 56 | 3e | 4b | c6 | d2 | 79 | 20 | 9a | db | c0 | fe | 78 | cd | 5a | f4 |
| c | 1f | dd | a8 | 33 | 88 | 07 | c7 | 31 | b1 | 12 | 10 | 59 | 27 | 80 | ec | 5f |
| d | 60 | 51 | 7f | a9 | 19 | b5 | 4a | 0d | 2d | e5 | 7a | 9f | 93 | c9 | 9c | ef |
| e | a0 | e0 | 3b | 4d | ae | 2a | f5 | b0 | c8 | eb | bb | 3c | 83 | 53 | 99 | 61 |
| f | 17 | 2b | 04 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

Аффинное преобразование

$$cd = \begin{bmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \end{bmatrix} \times ab^{-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

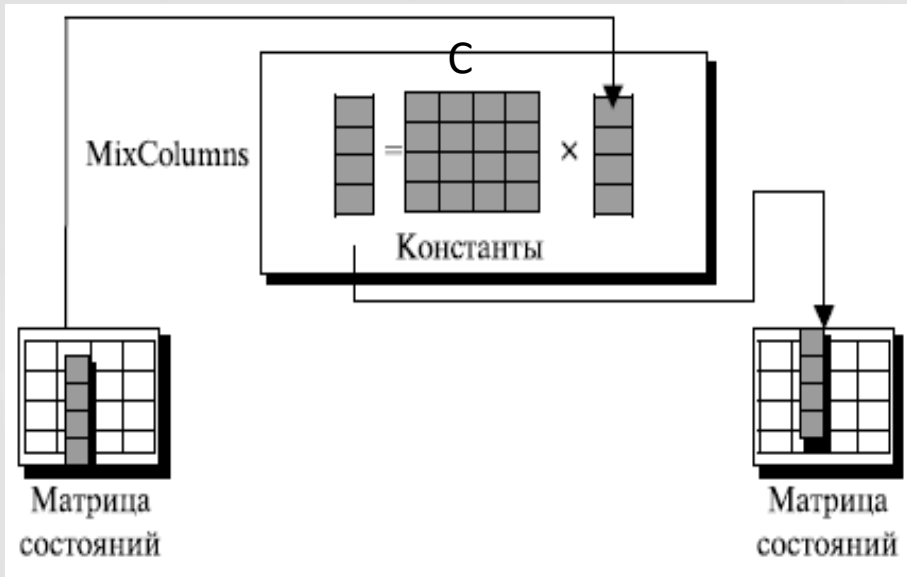
$$(ab) \times (ab^{-1}) \equiv 1 \bmod (x^8 + x^4 + x^3 + x + 1)$$

Преобразование «Перестановка»



- Операция ShiftRows производит циклический сдвиг влево
- Число сдвигов равно номеру обрабатываемой строки матрицы состояний
- Порядок битов в байте не меняется
- Для дешифрации используется процедура InvShiftRows – инверсная по отношению ShiftRows

Преобразование «Смешивание»



- *MixColumns* преобразовывает каждый столбец матрицы состояний в новый столбец умножением на матрицу констант C
- Столбцы матрицы состояний рассматриваются как полиномы с коэффициентами в $GF(2^8)$ и умножаются по модулю x^4+1 на полином:

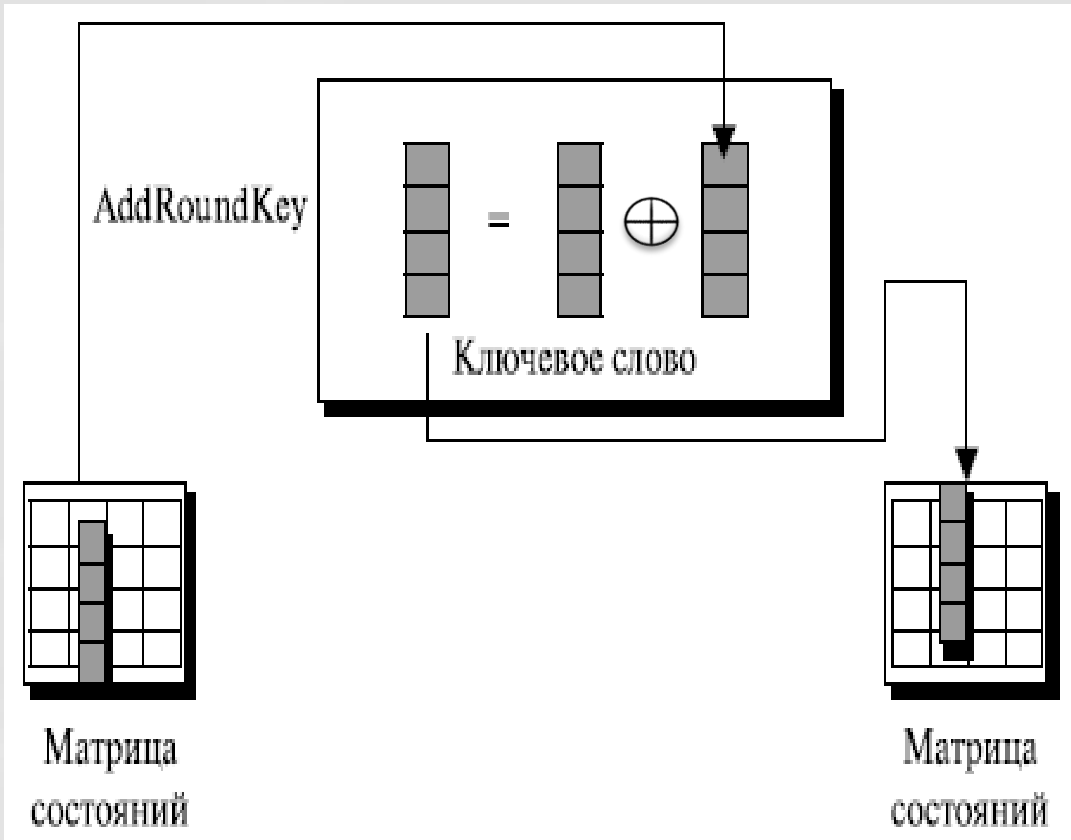
$$'03'x^3 + '01'x^2 + '01'x + '02'$$

- Матрица констант C является циклической и реализует умножение полиномов по модулю x^4+1
- Для дешифрации используется процедура *InvMixColumns* (с матрицей констант C^{-1}) – инверсная по отношению *MixColumns*

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \xleftrightarrow{\text{Инверсия}} \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

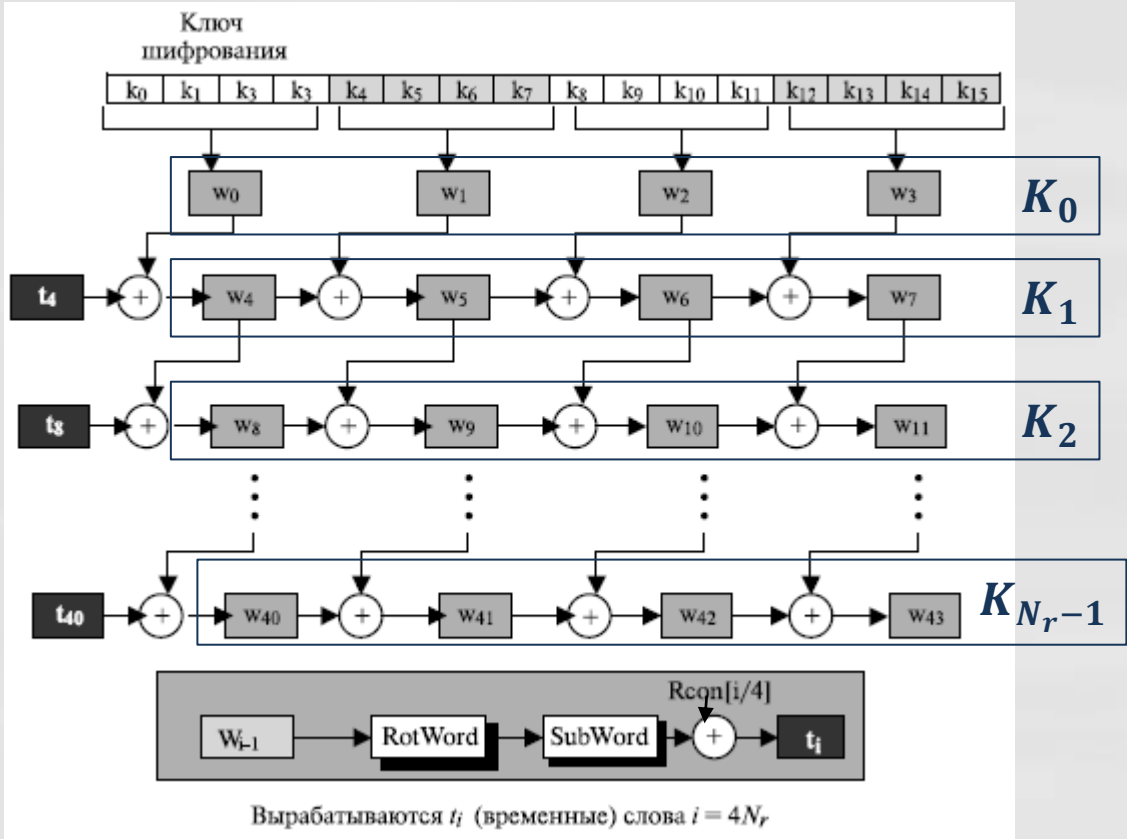
C C^{-1}

Преобразование «Сложение с раундовым ключом»



- Ключ раунда рассматривается как четыре слова по 32 бита
- Операция *AddRoundKey* выполняет XOR каждой колонки матрицы состояний с соответствующим словом раундового ключа
- Процедура *AddRoundKey* инверсна сама себе

Генерация (расширение множества) ключей

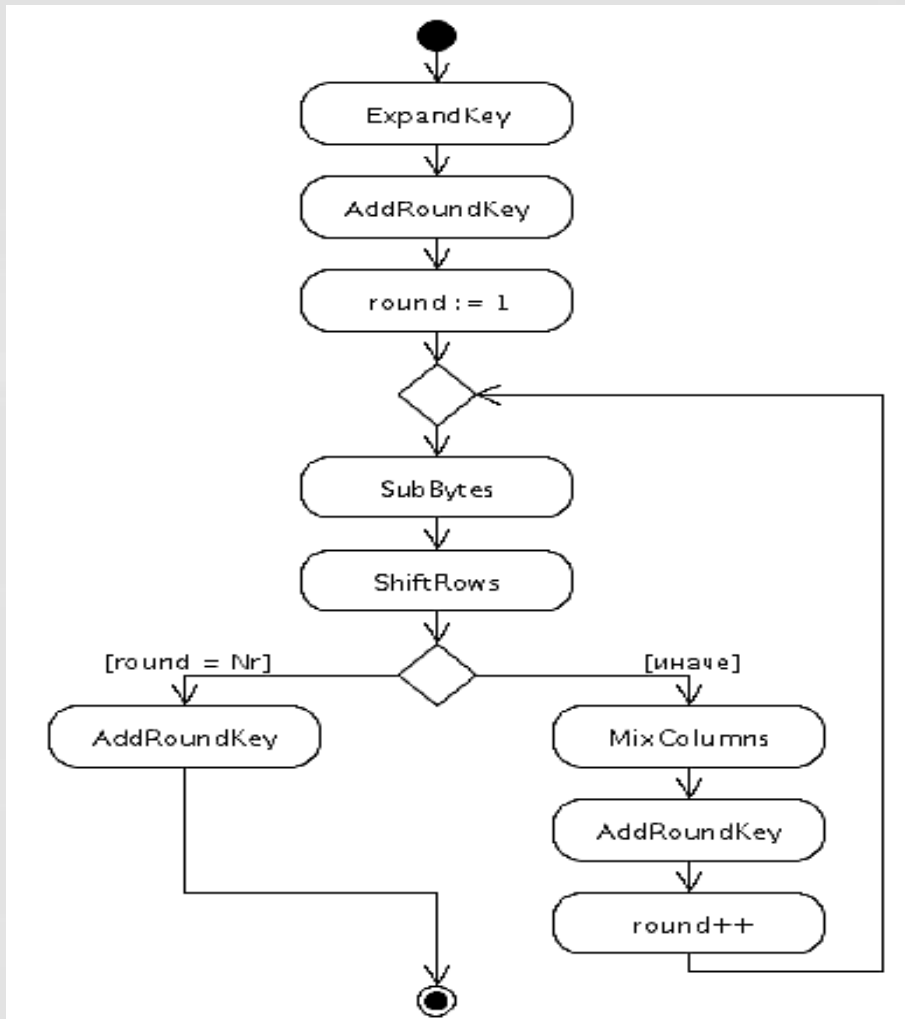


Пример генерации раундовых ключей для AES-128

- Процедура расширения ключей *ExpandKey* создает последовательно (слово за словом) 128-битные ключи N_r раундов от единственного ключа шифра
- RotWord* осуществляет циклический сдвиг байтов в слове влево
- SubWord* осуществляет замену каждого байта в слове в соответствии с методом подстановок, использованном в *SubByte*.
- Rcon(i)* формирует константу, i - номер раунда

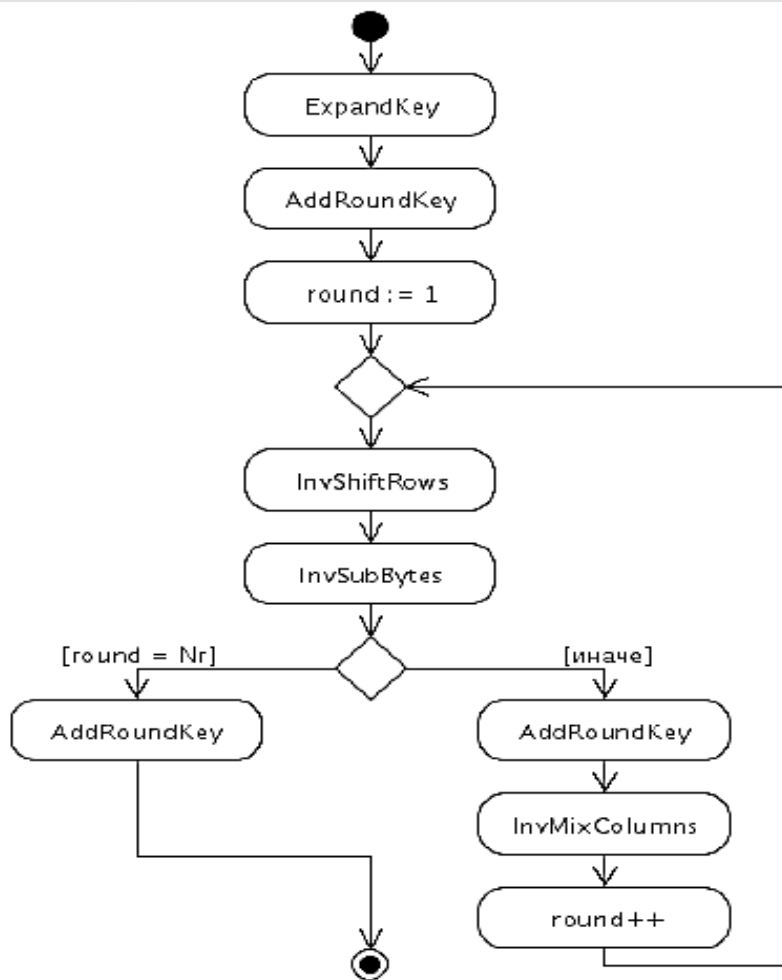
| Раунд | Константа (RCon) | Раунд | Константа (RCon) |
|-------|-------------------------|-------|-------------------------|
| 1 | $(01\ 00\ 00\ 00)_{16}$ | 6 | $(20\ 00\ 00\ 00)_{16}$ |
| 2 | $(02\ 00\ 00\ 00)_{16}$ | 7 | $(40\ 00\ 00\ 00)_{16}$ |
| 3 | $(04\ 00\ 00\ 00)_{16}$ | 8 | $(80\ 00\ 00\ 00)_{16}$ |
| 4 | $(08\ 00\ 00\ 00)_{16}$ | 9 | $(1B\ 00\ 00\ 00)_{16}$ |
| 5 | $(10\ 00\ 00\ 00)_{16}$ | 10 | $(36\ 00\ 00\ 00)_{16}$ |

Зашифрование



- Перед первым раундом применяется дополнительное забеливание с использованием ключа. Без этого любой слой до первого добавления ключа может быть просто снят без знания ключа, что не добавляет безопасности в алгоритм
- Для того чтобы сделать структуру алгоритма более простой, слой линейного перемешивания столбцов в последнем раунде отсутствует. Доказано, что это не повышает и не понижает безопасность.

Расшифрование

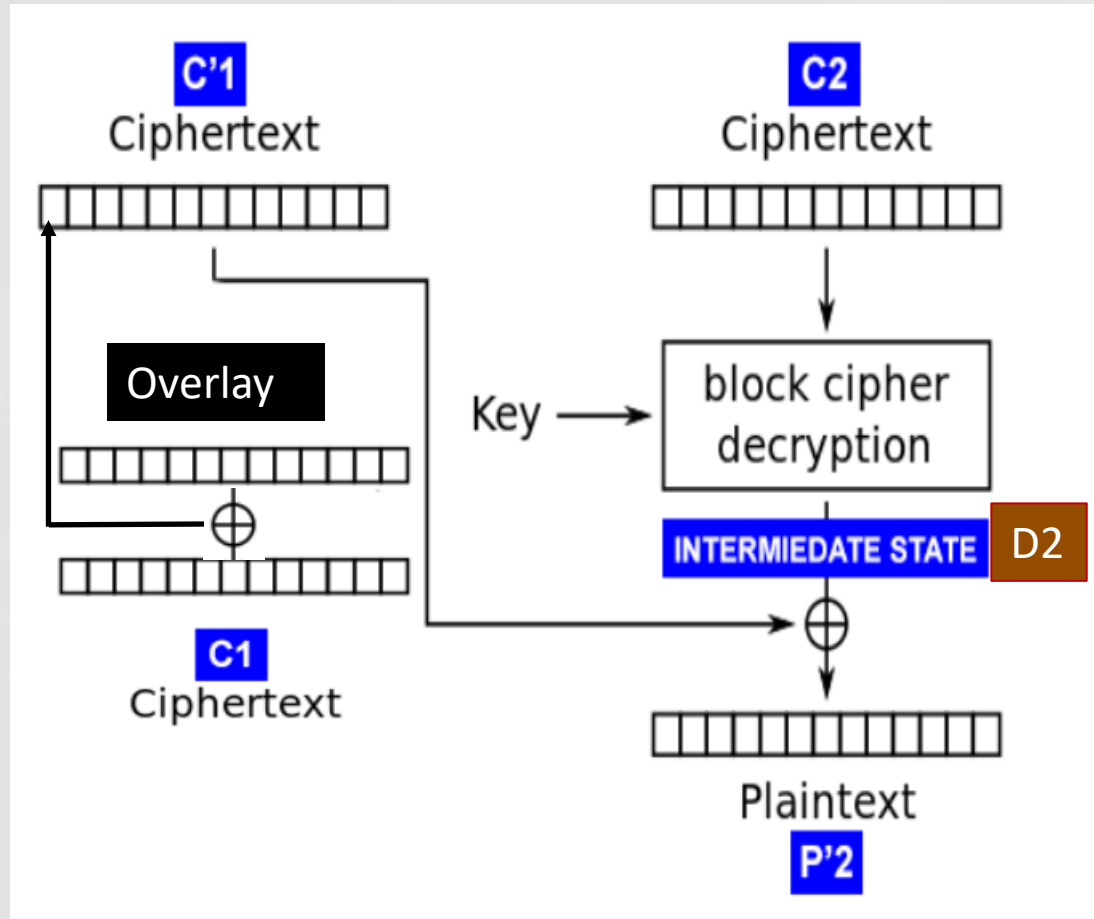


- При расшифровании все преобразования производятся в обратном порядке
- Используются следующие обратные преобразования вместо соответствующих шифрующих:
 - *InvSubBytes* — подстановка байтов с помощью обратной таблицы подстановок;
 - *InvShiftRows* — циклический сдвиг строк матрицы состояния на различные величины;
 - *InvMixColumns* — смешивание столбцов внутри каждого столбца матрицы состояния
- Ключи раундов используются в обратном порядке

Достоинства AES

- Обеспечивает быстрое перемешивание информации, при этом за один раунд преобразованию подвергается весь входной блок
- Байт ориентированная структура удобна для реализации на восьмиразрядных микроконтроллерах
- Все раундовые преобразования – это операции в конечных полях, допускающие эффективную аппаратную и программную реализацию на различных платформах
- Преобразование раунда допускает параллельное выполнение, что является важным преимуществом для будущих процессоров и специализированной аппаратуры

Padding Oracle Attack (CBC mode)



См. детали в

<https://habrahabr.ru/post/247527/>

- **Цель:** Расшифровать C_2 , зная правило дополнения блока открытого текста P_2
- **Метод:** Изменяем блок шифротекста, передаем измененный блок C'_1 и анализируем реакцию сервера-получателя на корректность дополнения:

$$P'_2 = D_2 \oplus C'_1$$

- Знание реакции сервера-получателя позволяет восстановить $D_2 = P'_2 \oplus C'_1$
- Расшифровка производится по правилу $P_2 = D_2 \oplus C_1$

МЕТОДЫ СИММЕТРИЧНОГО ШИФРОВАНИЯ

Алгоритм стандарта шифрования ГОСТ Р 34.12–2015,
ГОСТ Р 34.13–2015



Изменения стандарта в 2015

ГОСТ 28147-89

Системы обработки информации.
Защита криптографическая. Алгоритм
криптографического преобразования

ГОСТ Р 34.12–2015

"Информационная технология. Криптографическая
защита информации. Блочные шифры"
(Утвержден и введен в действие Приказом
Федерального агентства по техническому
регулированию и метрологии от 19 июня 2015г. №
749-ст)

ГОСТ Р 34.13–2015

"Информационная технология.
Криптографическая защита информации.
Режимы работы блочных шифров"
(Утвержден и введен в действие Приказом
Федерального агентства по техническому
регулированию и метрологии от 19 июня 2015г.
№ 750-ст)

Блочные шифры стандарта ГОСТ 34.12-15

Магма

Это ГОСТ 28147-89 с фиксированным набором подстановок

Узел замены, определенный Техническим комитетом по стандартизации "Криптографическая защита информации" (сокращенно - ТК 26) Росстандарта (Рекомендации по стандартизации "Задание узлов замены блока подстановки алгоритма шифрования ГОСТ 28147-89")

Кузнечик

Новый алгоритм, в основе которого SP-сеть: преобразование, состоящее из нескольких одинаковых раундов, при этом каждый раунд состоит из нелинейного и линейного преобразований, а также операции наложения ключа.

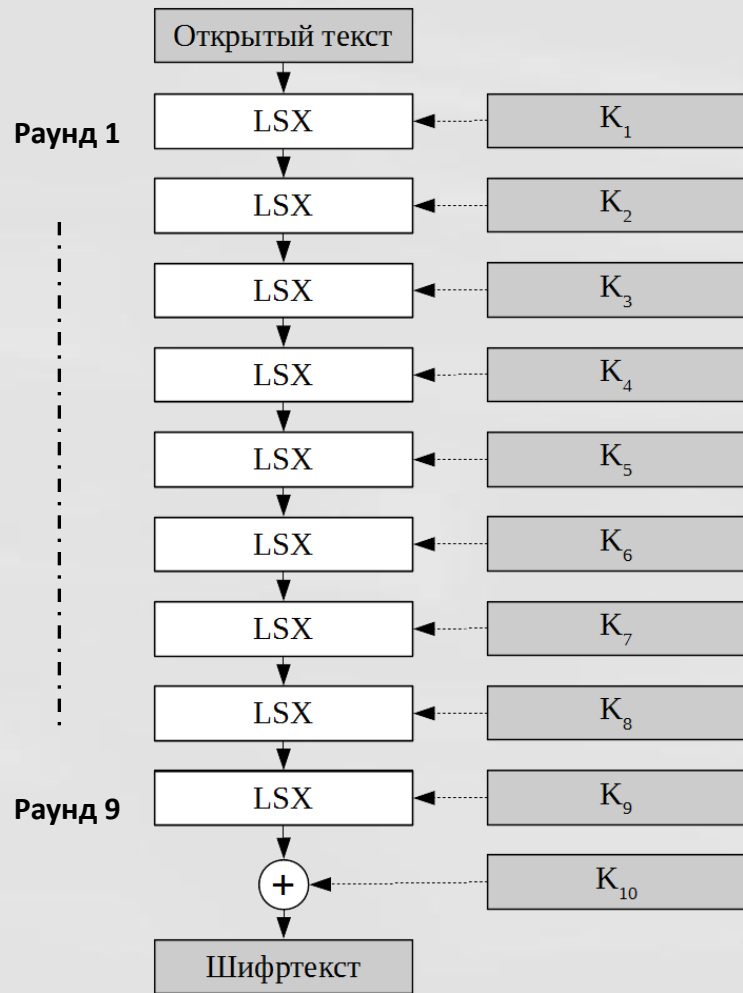
В отличие от алгоритма AES у «Кузнечика» есть ряд своих особенностей:

- линейное преобразование может быть реализовано с помощью регистра сдвига;
- Генерация раундовых ключей реализована с помощью сети Фейстеля, в которой в качестве функции используется раундовое преобразование алгоритма шифрования (Кузнечик)

S-блоки в шифре «Магма»

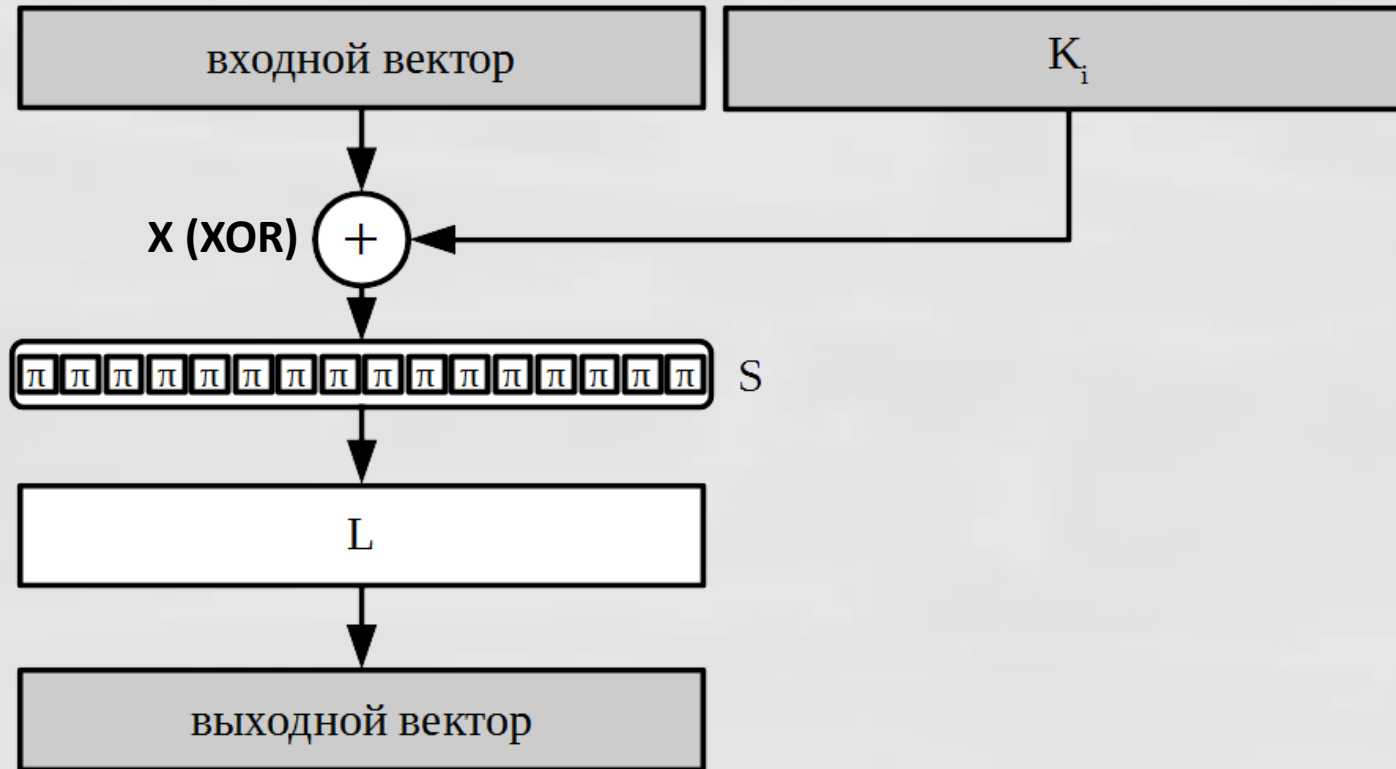
| Номер S- блока | Подстановка | | | | | | | | | | | | | | | |
|----------------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 1 | 12 | 4 | 6 | 2 | 10 | 5 | 11 | 9 | 14 | 8 | 13 | 7 | 0 | 3 | 15 | 1 |
| 2 | 6 | 8 | 2 | 3 | 9 | 10 | 5 | 12 | 1 | 14 | 4 | 7 | 11 | 13 | 0 | 15 |
| 3 | 11 | 3 | 5 | 8 | 2 | 15 | 10 | 13 | 14 | 1 | 7 | 4 | 12 | 9 | 6 | 0 |
| 4 | 12 | 8 | 2 | 1 | 13 | 4 | 15 | 6 | 7 | 0 | 10 | 5 | 3 | 14 | 9 | 11 |
| 5 | 7 | 15 | 5 | 10 | 8 | 1 | 6 | 13 | 0 | 9 | 3 | 14 | 11 | 4 | 2 | 12 |
| 6 | 5 | 13 | 15 | 6 | 9 | 2 | 12 | 10 | 11 | 7 | 8 | 1 | 4 | 3 | 14 | 0 |
| 7 | 8 | 14 | 2 | 5 | 6 | 9 | 1 | 12 | 15 | 4 | 11 | 0 | 13 | 10 | 3 | 7 |
| 8 | 1 | 7 | 14 | 13 | 0 | 5 | 8 | 3 | 4 | 15 | 10 | 6 | 9 | 12 | 11 | 2 |

Шифр «Кузнечик» ГОСТ 34.12-15



- SP-сеть с набором преобразований **L, S, X**
- Количество раундов - 9
- Длина блока – 128 бит
- Длина ключа – 256 бит
- Генерация (развертывание) раундовых ключей основана на сети Фейстеля

Раундовое преобразование

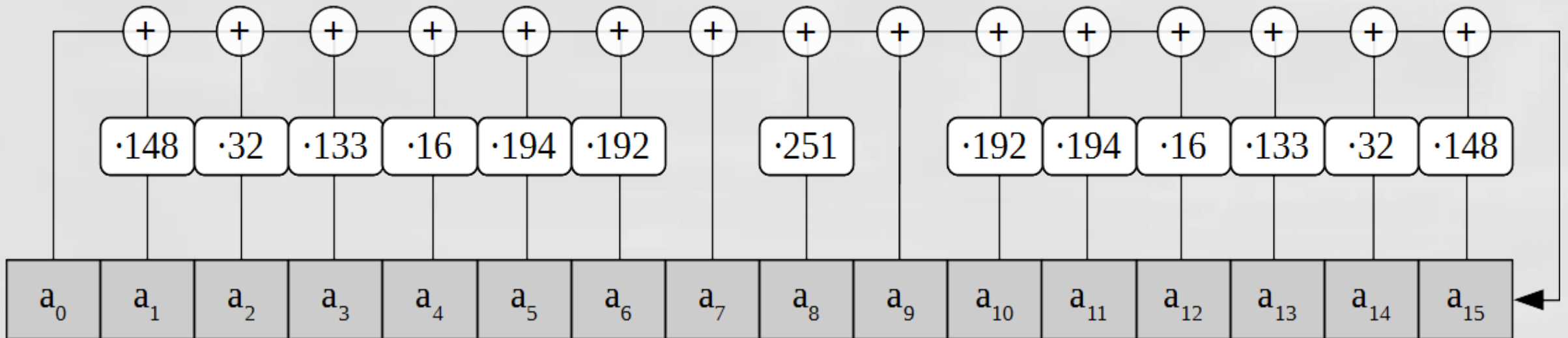


Подстановка S

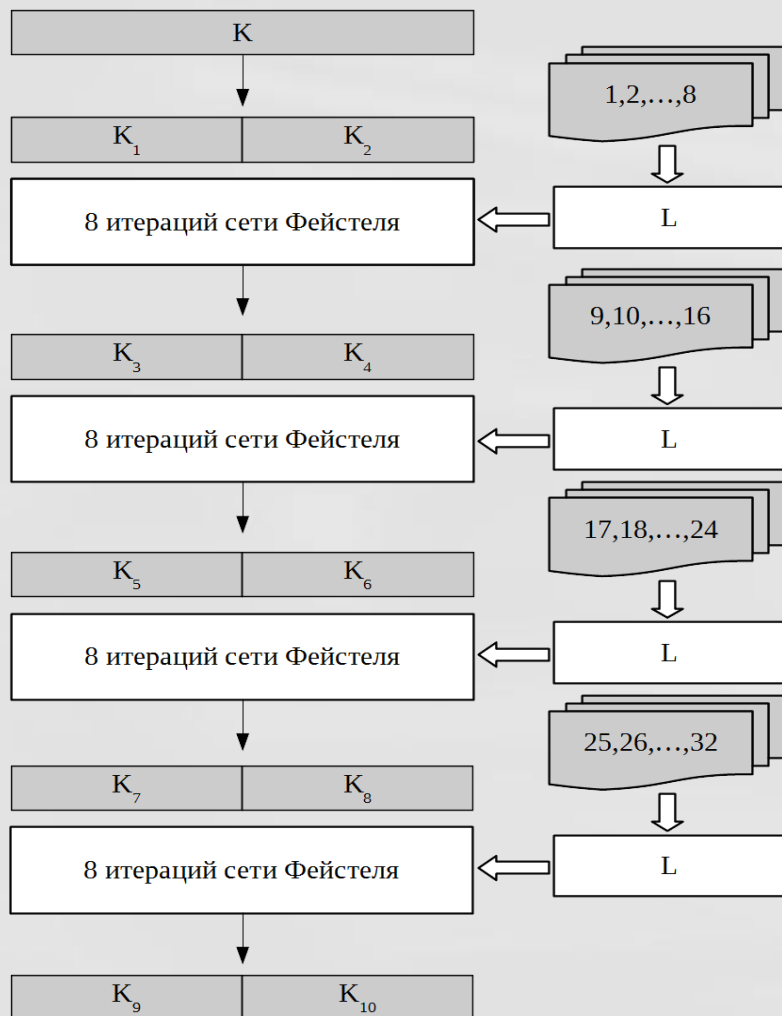
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 252 | 238 | 221 | 17 | 207 | 110 | 49 | 22 | 251 | 196 | 250 | 218 | 35 | 19 | 4 | 77 |
| 1 | 233 | 119 | 240 | 219 |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

Линейное преобразование L

- L представимо в виде регистра сдвига с обратной связью, который движется 16 раз
- Регистр реализуется над полем Галуа по модулю неприводимого многочлена $x^8 + x^7 + x^6 + x + 1$

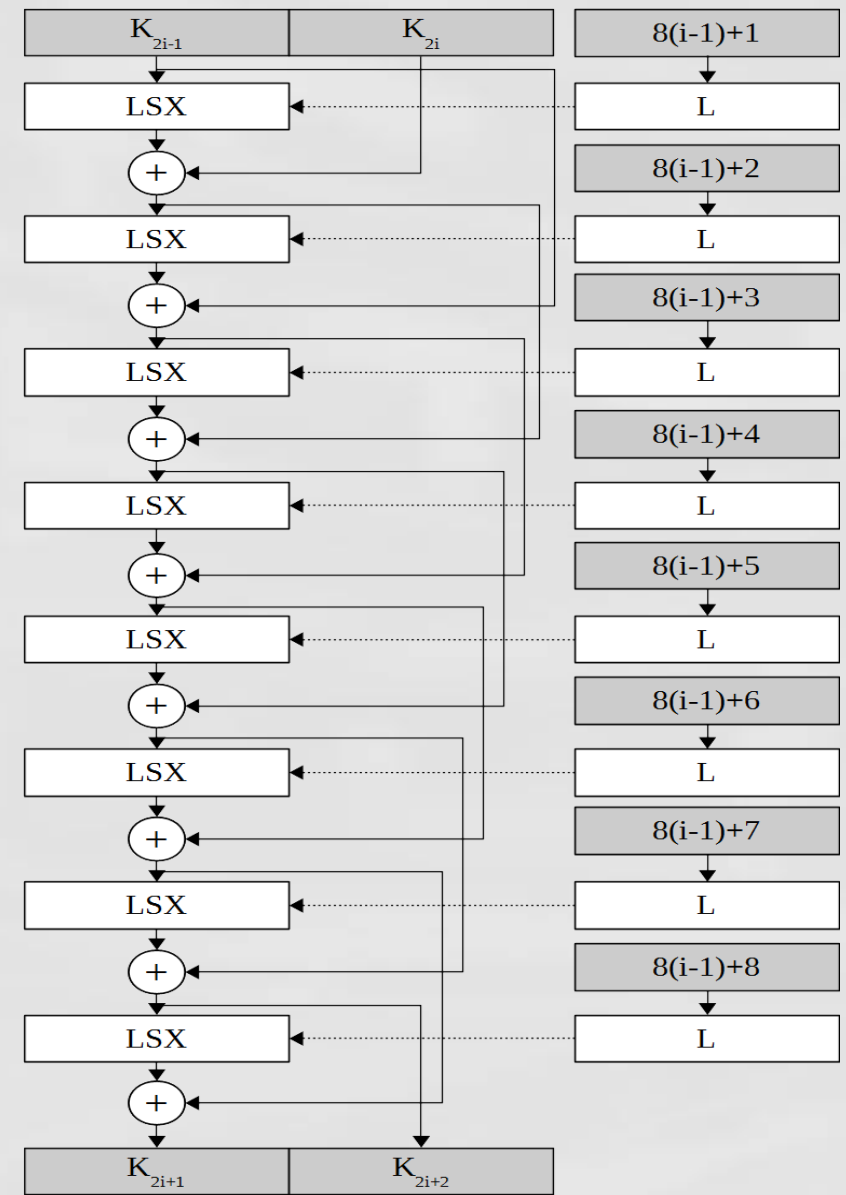
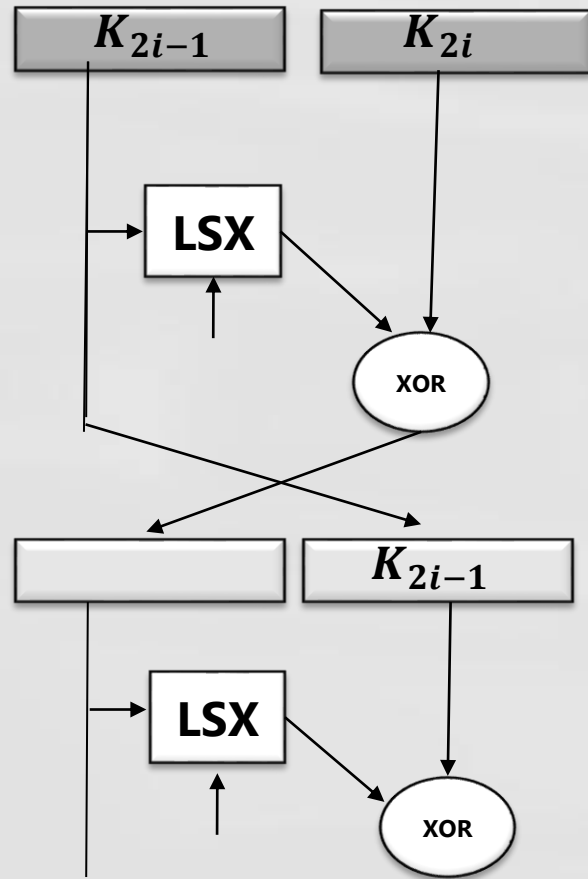


Генерация раундовых ключей



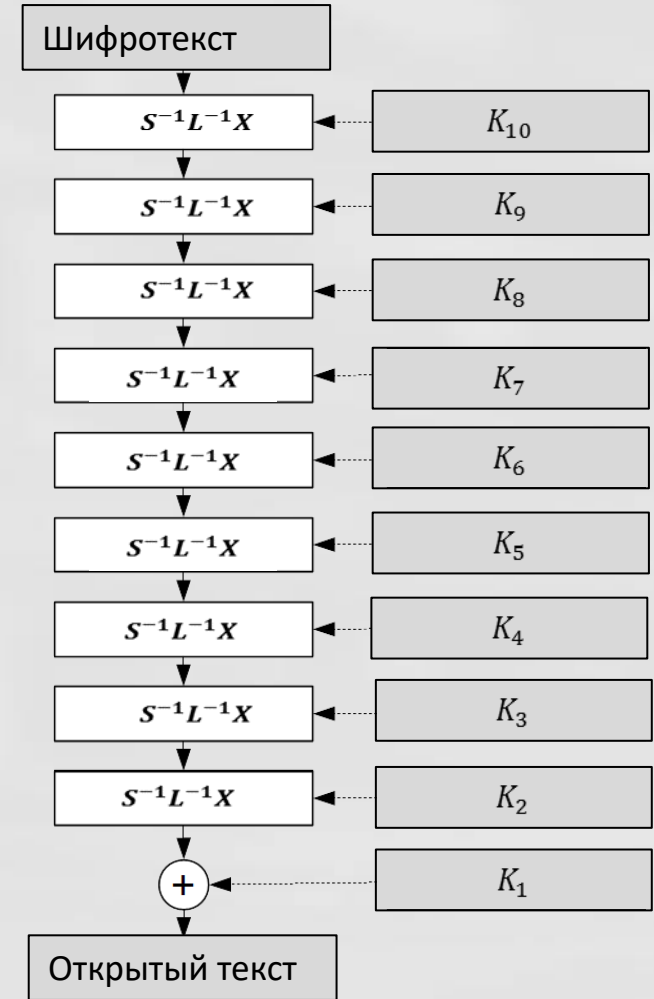
- Первые 2 ключа являются разбиением пополам основного ключа
- Каждая пара следующих ключей определяется с помощью пары предыдущих, прошедших через 8 раундов сети Фейстеля

Генерация i -той пары ключей, $i = 1, \dots, 4$

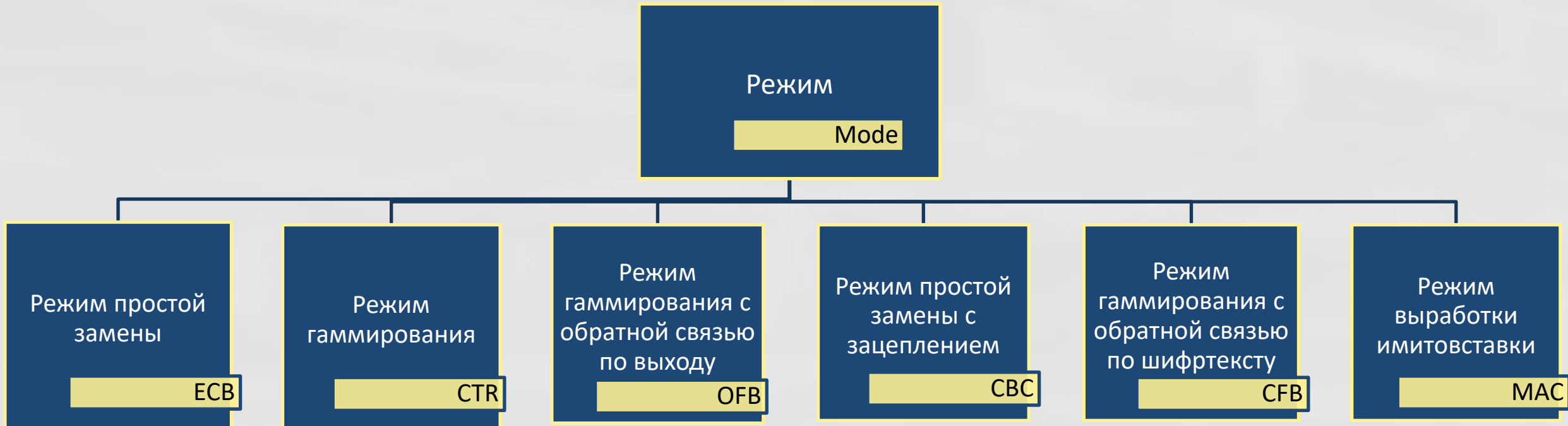


Расшифрование

- Расшифрование реализуется применением преобразований инверсным к базовым
- Раундовые ключи применяются в обратном порядке



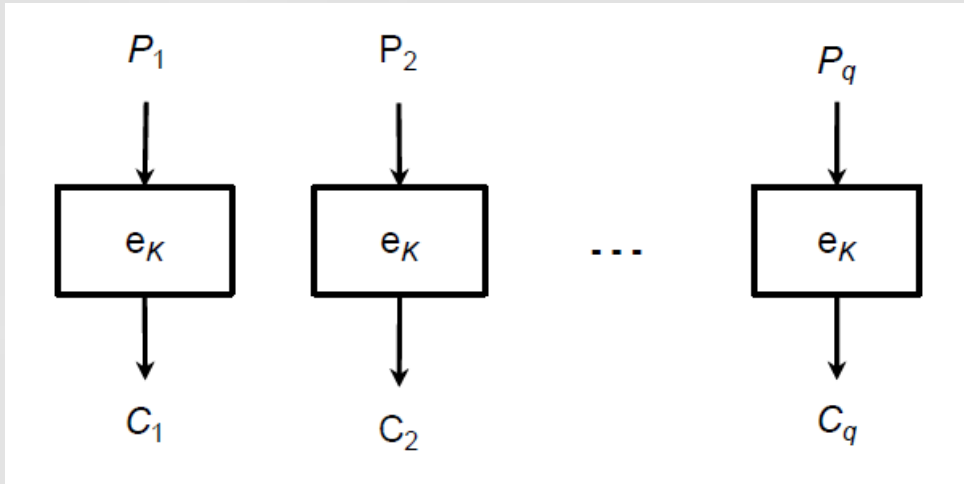
Режимы работы блочных шифров



Процедуры дополнения

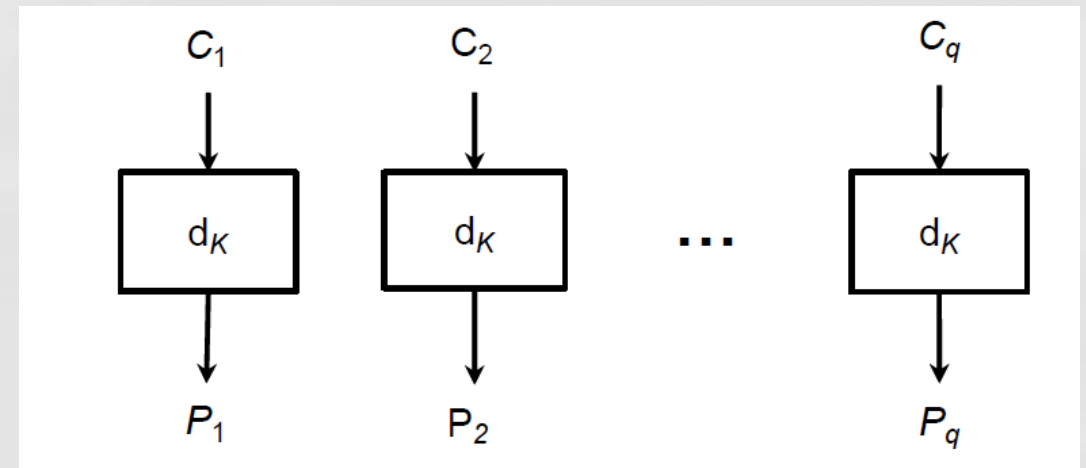
- Процедура 1: дополняем «00000000...0» (нулями). В общем случае получателю нужно знать длину сообщения.
- Процедура 2: дополняем «100000000...0» (единица с нулями)
- Процедура 3: НЕ дополняем, если длина сообщения кратна размеру блока, а иначе дополняем в соответствии с Процедурой (обязательна для режима выработки имитовставки и не рекомендуется для других режимов)

Режим простой замены

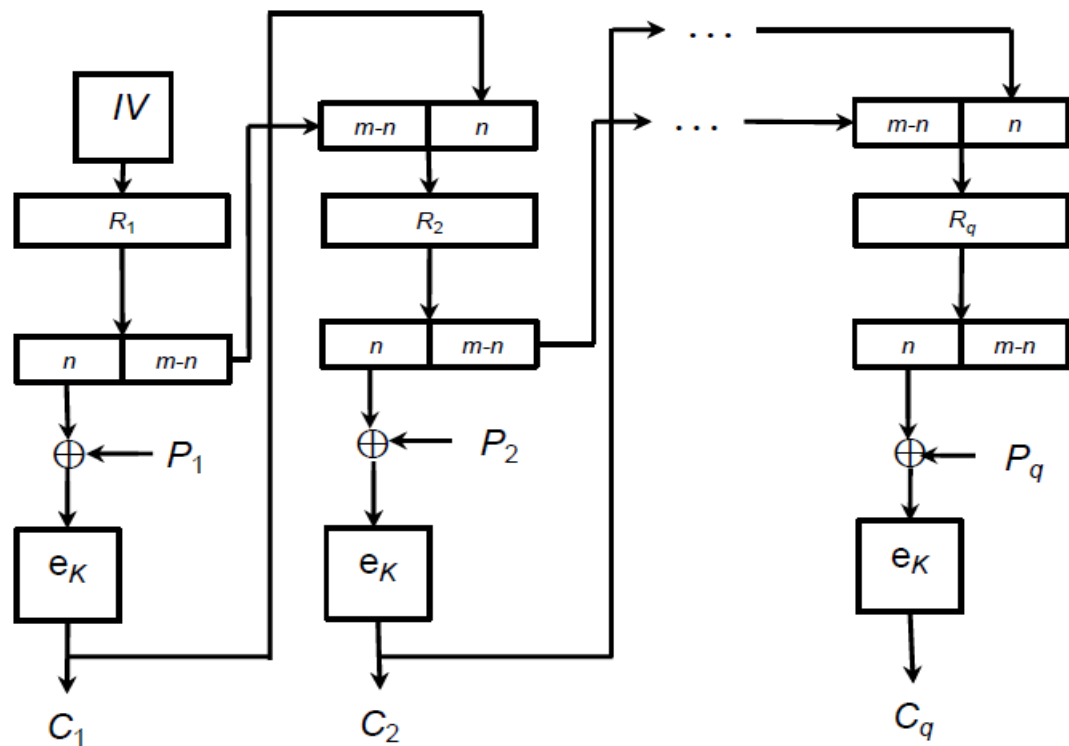


Electronic Codebook, ECB

ECB является самым простым и самым быстрым способом использования блочного шифра. ECB хорошо использовать для шифрования случайных данных, например, других ключей.



Режим простой замены с зацеплением



$$R_1 = IV,$$

$$\begin{cases} C_i = e_K(P_i \oplus \text{MSB}_n(R_i)), \\ R_{i+1} = \text{LSB}_{m-n}(R_i) \parallel C_i, \end{cases} \quad i = 1, 2, \dots, q-1,$$

$$C_q = e_K(P_q \oplus \text{MSB}_n(R_q)).$$

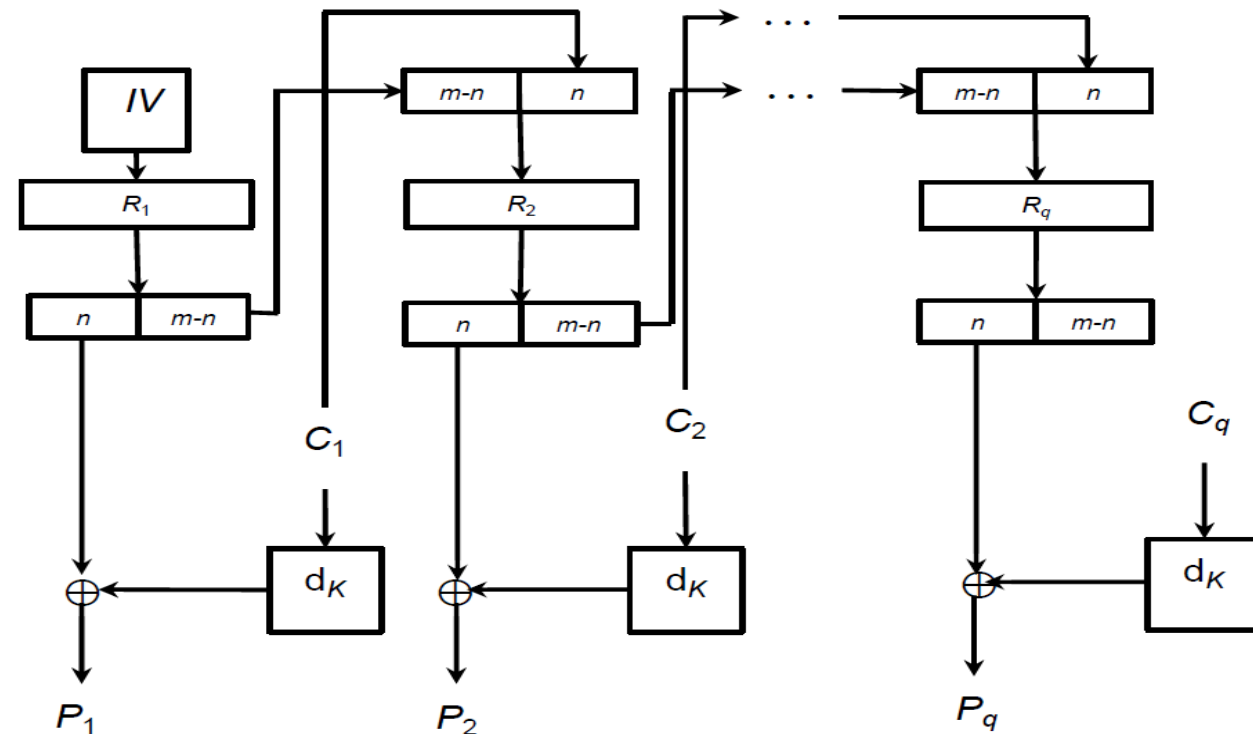
Cipher Block Chaining, CBC

CBC лучше всего подходит для шифрования файлов

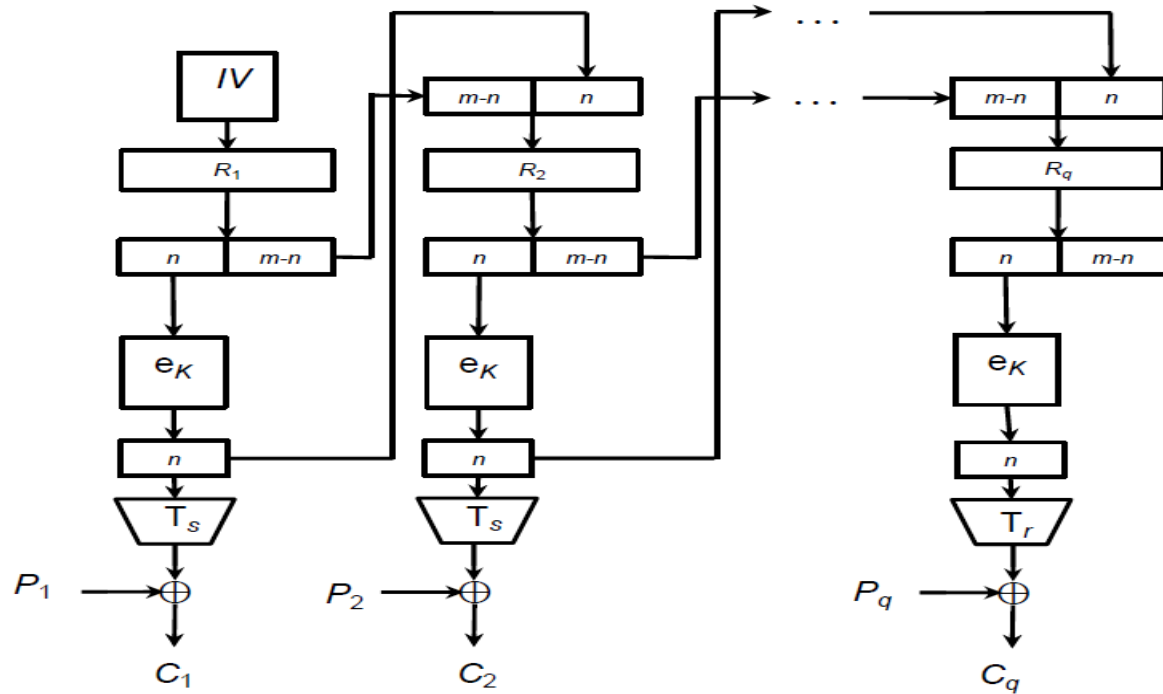
$$R_1 = IV,$$

$$\begin{cases} P_i = d_K(C_i) \oplus \text{MSB}_n(R_i), \\ R_{i+1} = \text{LSB}_{m-n}(R_i) \parallel C_i, \end{cases} \quad i = 1, 2, \dots, q-1,$$

$$P_q = d_K(C_q) \oplus \text{MSB}_n(R_q).$$



Режим гаммирования с обратной связью по выходу



$$R_1 = IV,$$

$$\begin{cases} Y_i = e_K(\text{MSB}_n(R_i)), \\ C_i = P_i \oplus T_s(Y_i), \\ R_{i+1} = \text{LSB}_{m-n}(R_i) \parallel Y_i, \end{cases} \quad i = 1, 2, \dots, q-1,$$

$$Y_q = e_K(\text{MSB}_n(R_q)),$$

$$C_q = P_q \oplus T_r(Y_q).$$

**Output
Feedback, OFB**

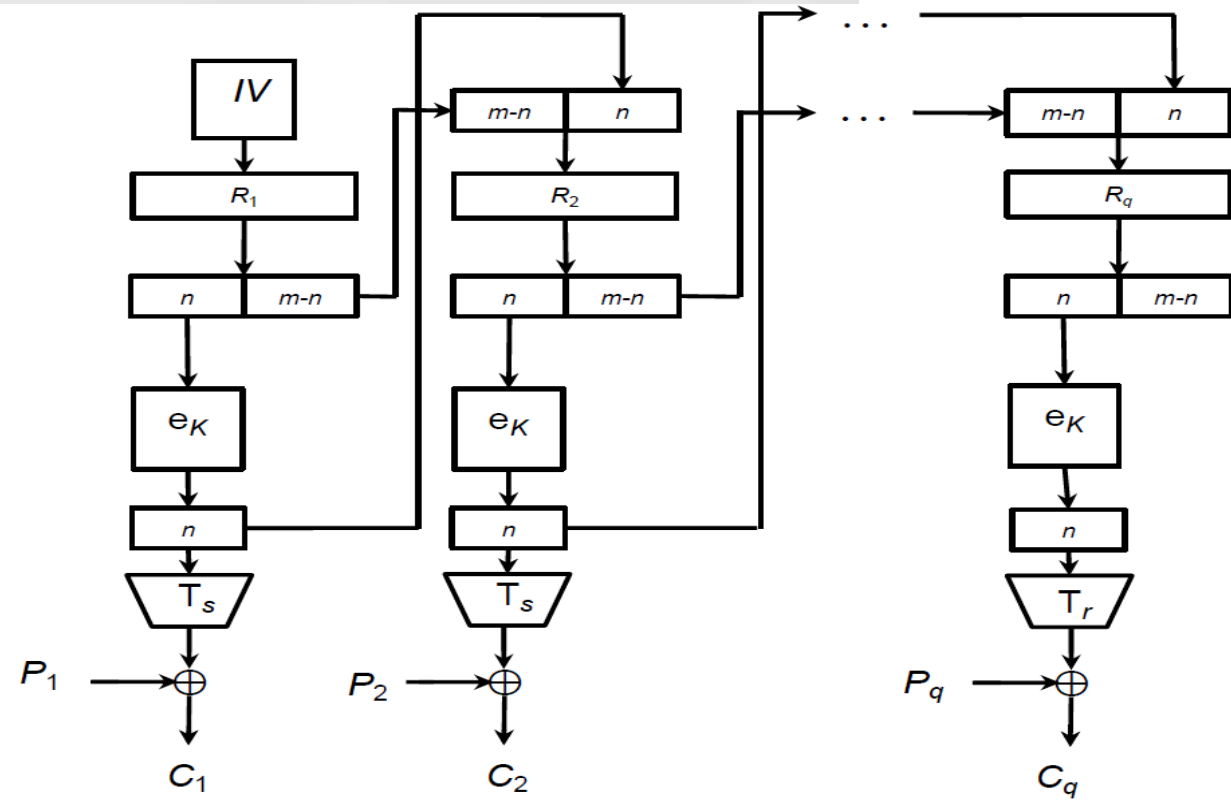
OFB чаще всего используется в высокоскоростных синхронных системах, где недопустимо распространение ошибки

$$R_1 = IV,$$

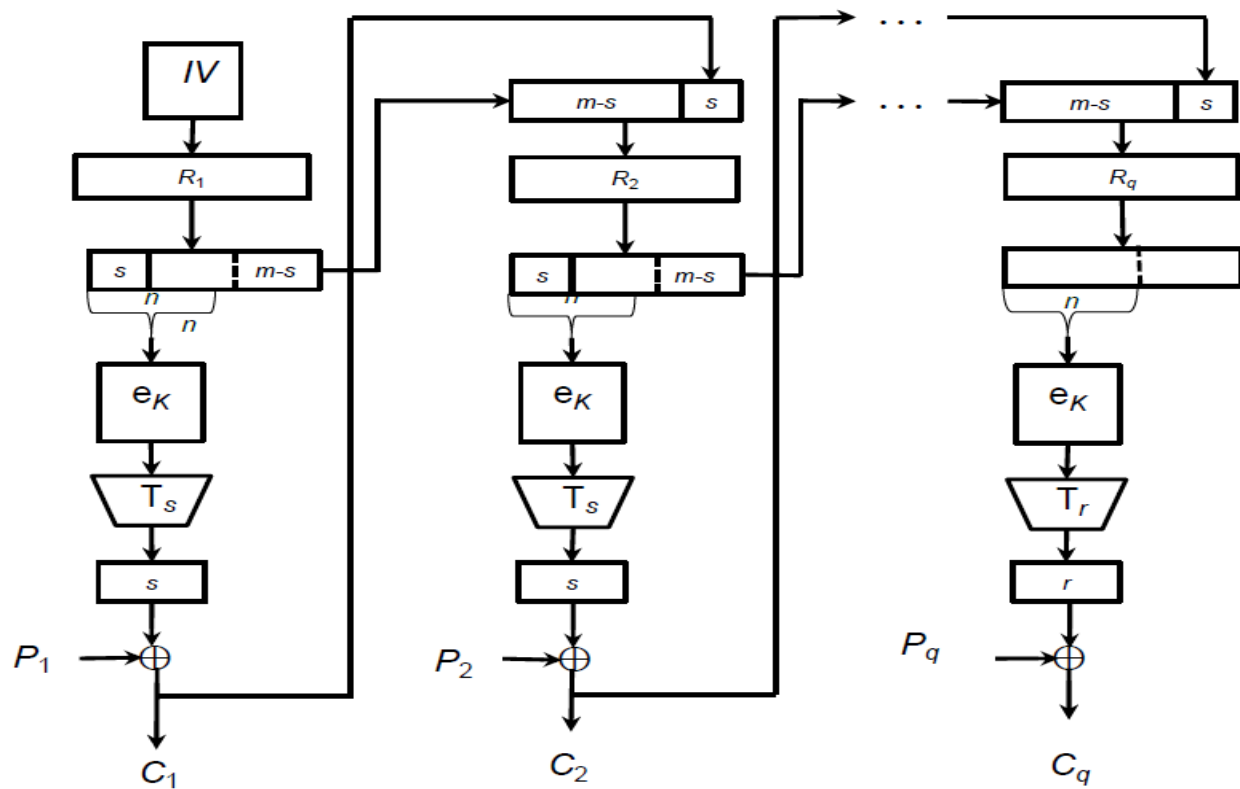
$$\begin{cases} Y_i = e_K(\text{MSB}_n(R_i)), \\ P_i = C_i \oplus T_s(Y_i), \\ R_{i+1} = \text{LSB}_{m-n}(R_i) \parallel Y_i, \end{cases}$$

$$Y_q = e_K(\text{MSB}_n(R_q)),$$

$$P_q = C_q \oplus T_r(Y_q)$$



Режим гаммирования с обратной связью по шифротексту



$$R_1 = IV,$$

$$\begin{cases} C_i = P_i \oplus T_s(e_K(\text{MSB}_n(R_i))), \\ R_{i+1} = \text{LSB}_{m-s}(R_i) \parallel C_i, \end{cases} \quad i = 1, 2, \dots, q-1,$$

$$C_q = P_q \oplus T_r(e_K(\text{MSB}_n(R_q))).$$

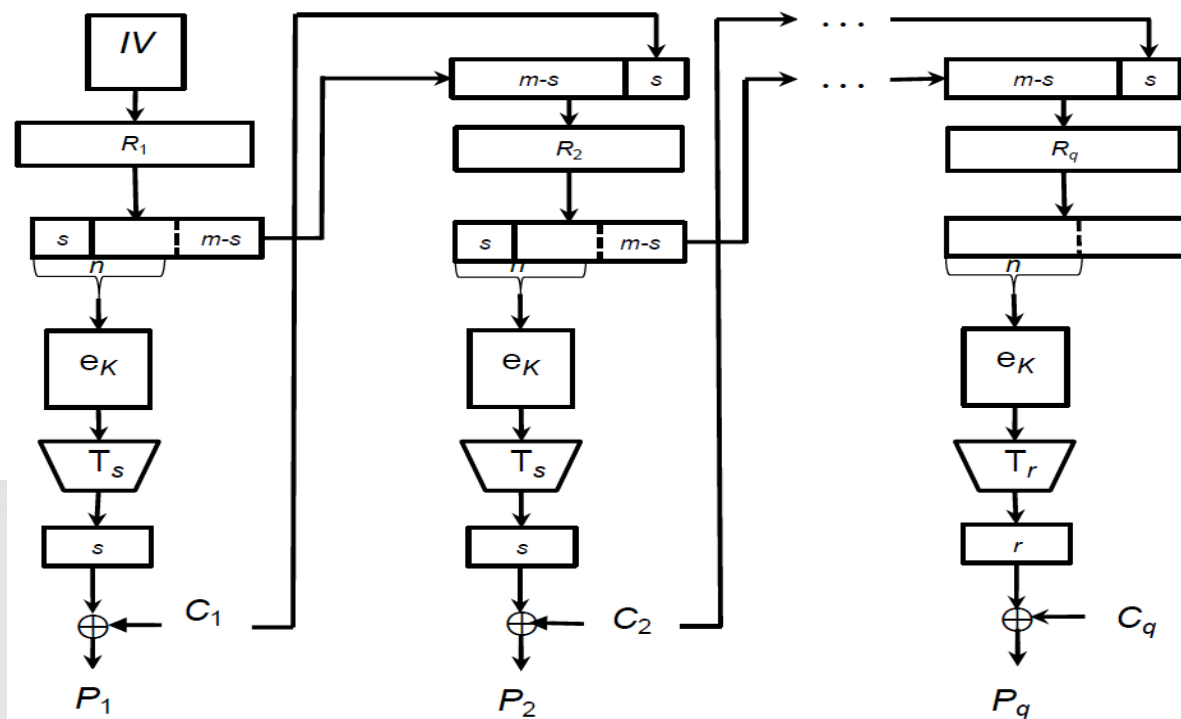
**Cipher
Feedback, CFB**

CFB обычно выбирается для шифрования потока символов, когда каждый символ может рассматриваться отдельно, как в линии связи между терминалом и главным компьютером

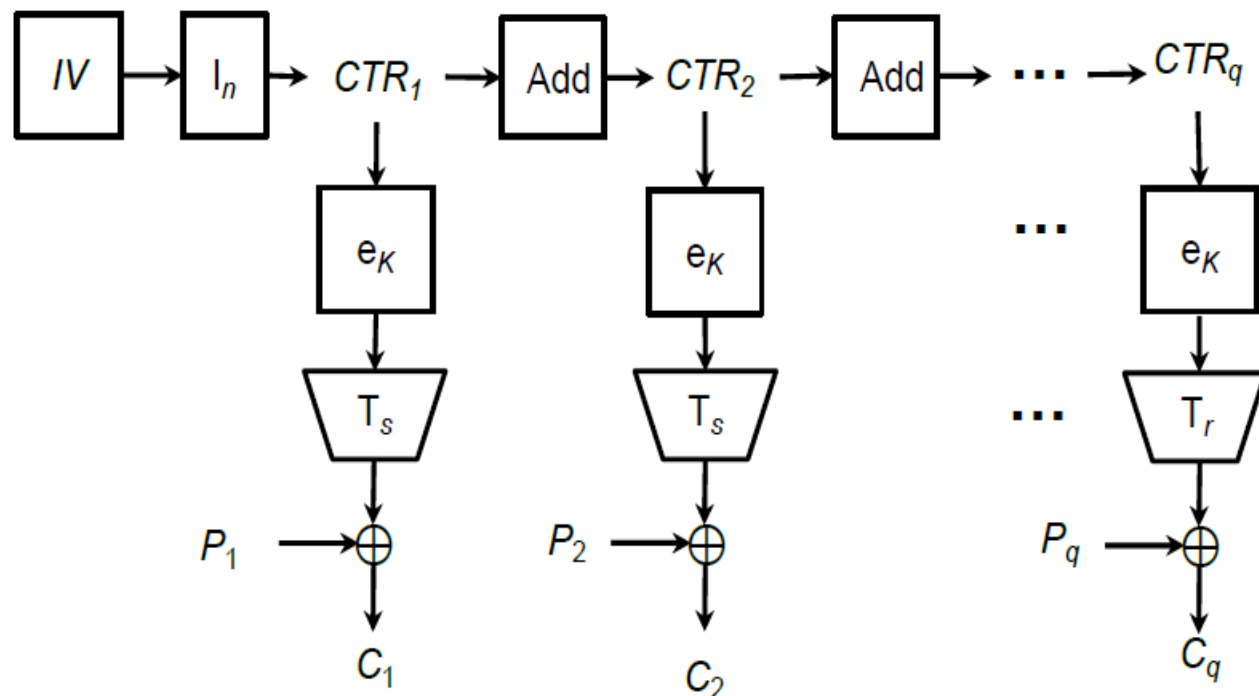
$$R_1 = IV,$$

$$\begin{cases} P_i = C_i \oplus T_s(e_K(\text{MSB}_n(R_i))), \\ R_{i+1} = \text{LSB}_{m-s}(R_i) \parallel C_i, \end{cases}$$

$$P_q = C_q \oplus T_r(e_K(\text{MSB}_n(R_q))).$$



Режим гаммирования



$$C_i = P_i \oplus T_s(e_K(CTR_i)), \quad i = 1, 2, \dots, q-1,$$

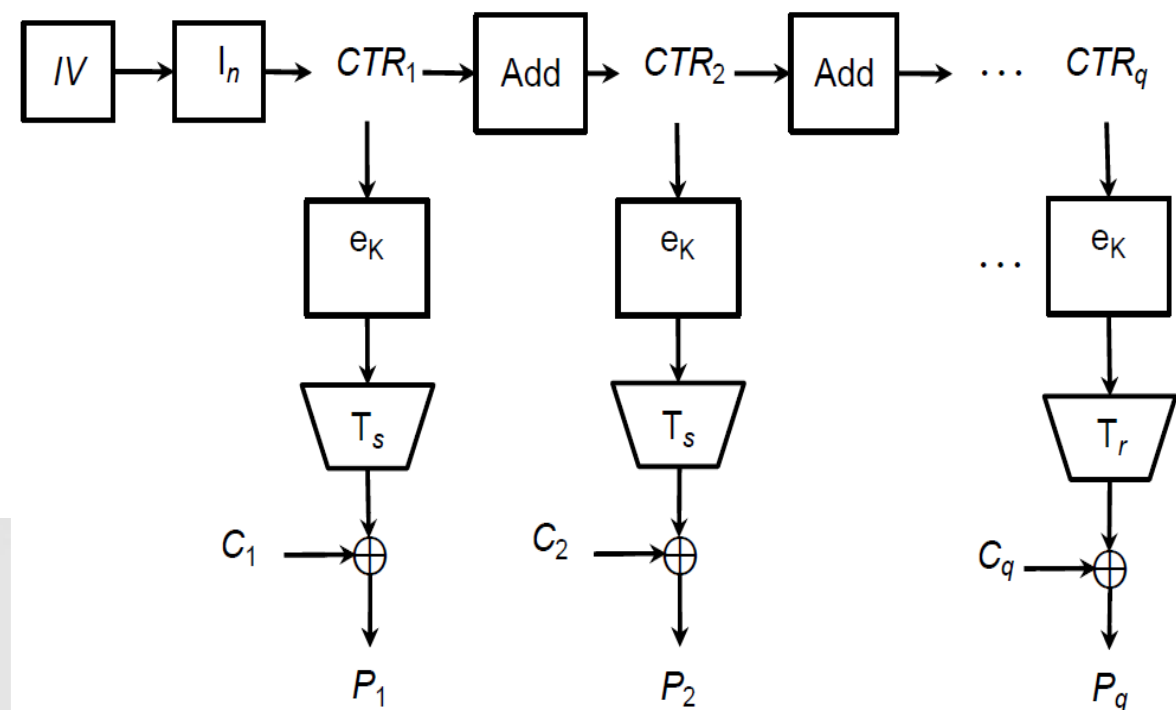
$$C_q = P_q \oplus T_r(e_K(CTR_q)).$$

Counter, CTR

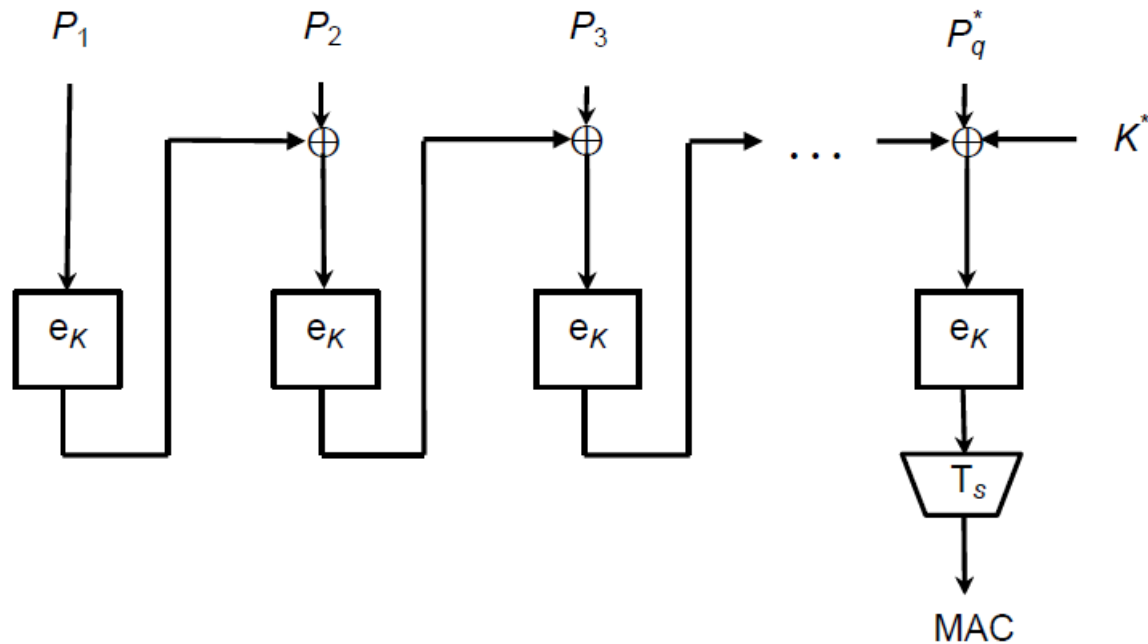
Для CRT свойства
синхронизации и
распространения ошибки
такие же, как и для OFB

$$P_i = C_i \oplus T_s(e_K(CTR_i)),$$

$$P_q = C_q \oplus T_r(e_K(CTR_q)).$$



Режим выработки имитовставки



Используется для контроля целостности сообщения, а не для обеспечения сокрытия содержания

$$\begin{aligned} C_0 &= 0^n, \\ C_i &= e_K(P_i \oplus C_{i-1}), i = 1, 2, \dots, q-1, \\ MAC &= T_s(e_K(P_q^* \oplus C_{q-1} \oplus K^*)), \end{aligned}$$

$$K^* = \begin{cases} K_1, & \text{если } |P_q| = n, \\ K_2, & \text{иначе,} \end{cases}$$

Message Authentication Code (MAC) algorithm

Сайт ТК по стандартизации →

<https://tc26.ru/standarts/nationalnye-standarty/>

Текст стандарта шифров

https://tc26.ru/standard/gost/GOST_R_3412-2015.pdf →

Эталонная реализация шифров



ТЕХНИЧЕСКИЙ КОМИТЕТ ПО СТАНДАРТИЗАЦИИ «КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ»

Новости

Документы

Проекты документов ²

Активности

О нас

Форум

Q ≡ EN

Национальные стандарты



ГОСТ Р 34.10–2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи»

Утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 7 августа 2012г. № 215-ст



ГОСТ Р 34.11–2012 «Информационная технология. Криптографическая защита информации. Функция хэширования»

Утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 7 августа 2012г. № 216-ст



ГОСТ Р 34.12–2015 «Информационная технология. Криптографическая защита информации. Блочные шифры»

Утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 19 июня 2015г. № 749-ст



ГОСТ Р 34.13–2015 «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров»

Утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 19 июня 2015г. № 750-ст

https://tc26.ru/standard/gost/PR_GOSTR-bch_v5.zip

Обновление стандартов в 2018 г.

- ГОСТ 34.12-2018 – Информационная технология.
Криптографическая защита информации . Блочные шифры
- ГОСТ 34.13-2018 - Информационная технология.
Криптографическая защита информации. Режимы работы блочных шифров

***ОСНОВАНИЕ - терминологическая и
концептуальная увязка с международными
стандартами от ИСО***