# Лекция 6. Симметричное шифрование

Это алгоритмы, в которых отправитель и получатель используют один и тот же ключ.

## Особенности практического применения

Необходим выбор секретного ключа до начала обмена тайными сообщениями (должна состояться встреча, на которой участники обмена получат данный ключ)

Если ключи планируется менять, к данной встрече потребуется заготовить все ключи на запланированный срок секретного общения.

После получения ключа его нужно секретно:

- хранить,

- использовать,

- уничтожать (чтобы злоумышленник не смог использовать выброшенный ключ для прочтения старой корреспонденции).

## ГОСТ 28147-89

Ключ - 256 бит = 8 32-разрядных числа

31 раунд шифра Фейстеля

## ГОСТ 34.12-2015

Кузнечик

## Люцифер (IBM)

P – блок перестановки

S – блок подстановки

2 раунда

1973 и 1974 – Люцифер доработан

1975 – опубликован, получены замечания от АНБ (сократили длину ключа и нелинейных S-блоков)

## C:\Users\secur11\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\004.jpgData Encryption Standard (DES)

1973 – заказ на правительственный стандарт шифрования некритичной информации

Прототип системы шифрования, разработанной для IBM 3614

1977– принят NBS (теперь это NIST) как стандарт в США (АНБ не обнаружило математических и статистических изъянов)

Алгоритм использовался в системе межбанковских клиринговых расчетов и за неделю обрабатывал 560 000 сообщений на сумму в 1,5 трлн. долларов;

Был рассчитан на 10 лет

Ключ: 64 бит (8 байт)

Каждый бит – функция всех бит исходной последовательности и ключа

При длине ключа 56 бит число ключей-72\*10^15

Время расшифровки – 2^56 операций = 7\*10^16 операций

### Этапы шифрования DES

Преобразование ключа

Функция Фейстеля

Шифрование текста (использует функцию Фейстеля)

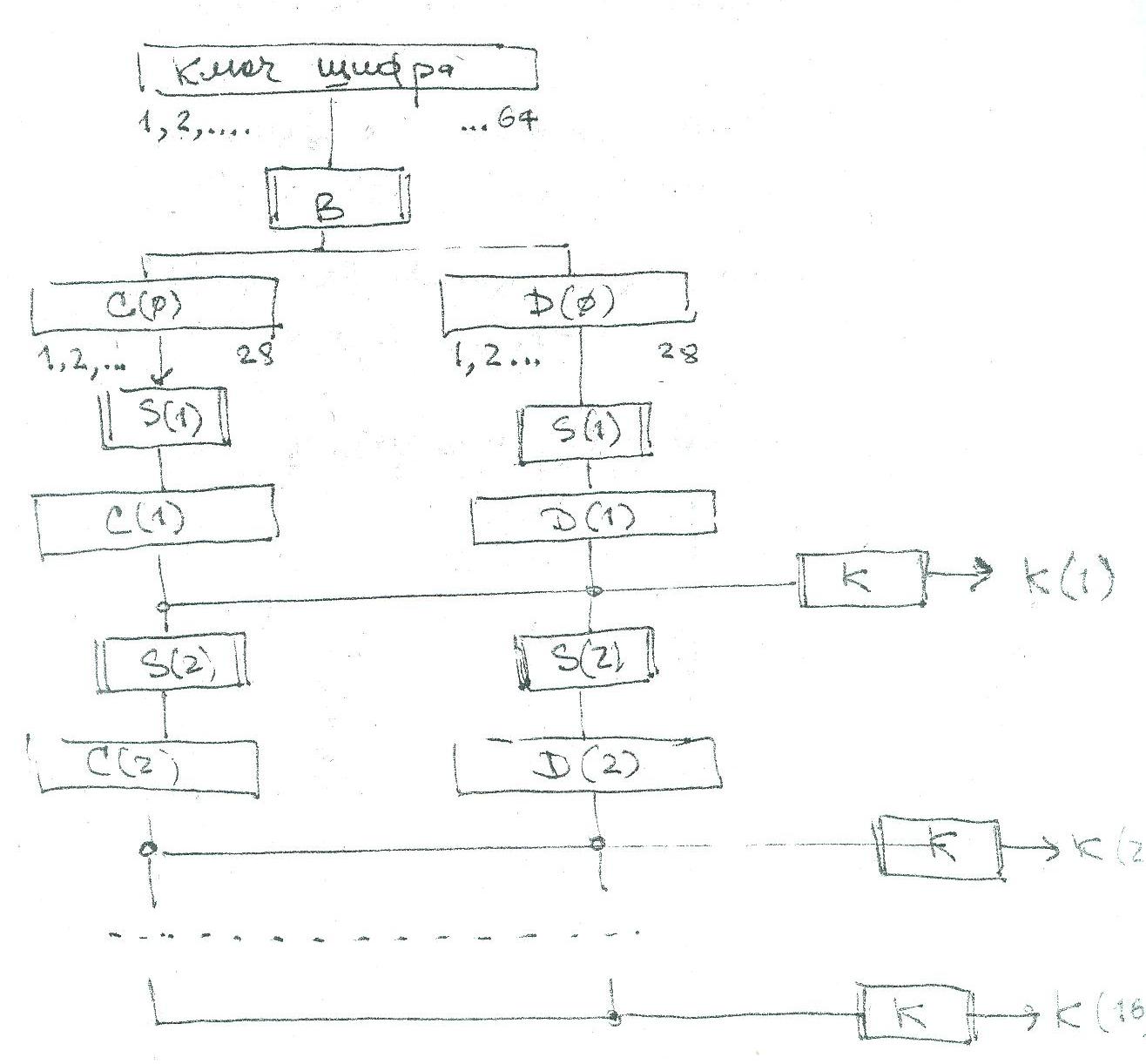
Расшифровка текста (использует функцию Фейстеля)

#### Преобразование ключа

64 бит -> 16 Х 48 бит

Алгоритм:

1. Переставить биты исходного ключа, исключая каждый 8-й бит (бит четности)
2. *Разделить на 2 половины по 28*
3. *Сдвинуть каждую половину на 1 или 2 бита*
4. *Объединить в 56 бит*
5. *Сжать до 48 бит*
6. *Повторить 16 раз с п.2*



Добавить к схеме сжимающую перестановку

Матрица перестановки B: результат – C(0)+D(0)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| С(0) | 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 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| D(0) | 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 |

Не используются биты с номерами, кратными восьми (8, 16….64 – биты четности)

S(i)-функция циклического сдвига влево (выполняется независимо для С(i) и D(i))

2,3,4….28,1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Итерация** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **Сдвиг** | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

К(i) – результат перестановки бит последовательности C(i)+D(i)

Для получения из 56 бит 48 применяется сжимающая перестановка Х?:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 14 | 17 | 11 | 24 | 1 | 5 | 3 | 28 | 15 | 6 | 21 | 10 | 23 | 19 | 12 | 4 |
| 26 | 8 | 16 | 7 | 27 | 20 | 13 | 2 | 41 | 52 | 31 | 37 | 47 | 55 | 30 | 40 |
| 51 | 45 | 33 | 48 | 44 | 49 | 39 | 56 | 34 | 53 | 46 | 42 | 50 | 36 | 29 | 32 |

Получаем ключи К1-К16

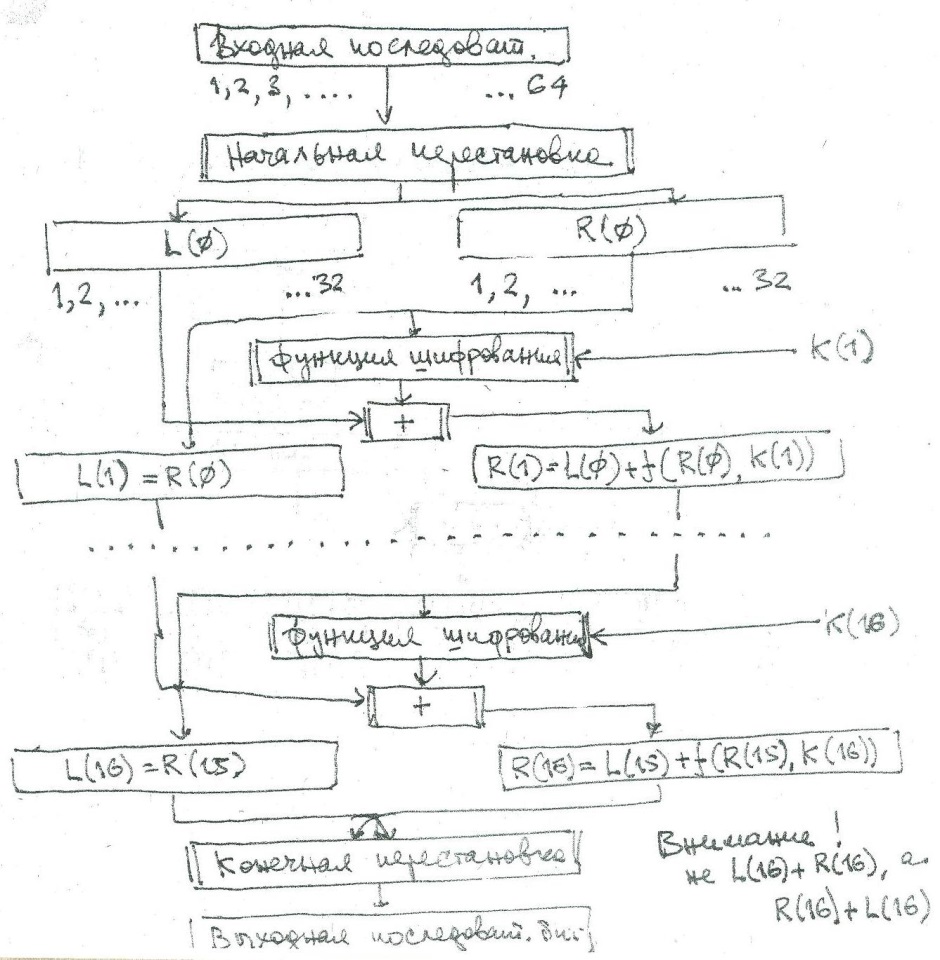
#### Шифрование текста

Алгоритм:

1. Начальная перестановка IP
2. Разделить на L и R
3. *L’=R*
4. *R’=L+f(R,Ki)*
5. *Повторить 16 раз с п.3*
6. Конечная перестановка IP-1

На каждом из 16 раундов:

1. L(i)=R(i-1)
2. R(i)=L(i-1)+F(R(i-1), K(i))



IP - Начальная перестановка для исходных 64 бит

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

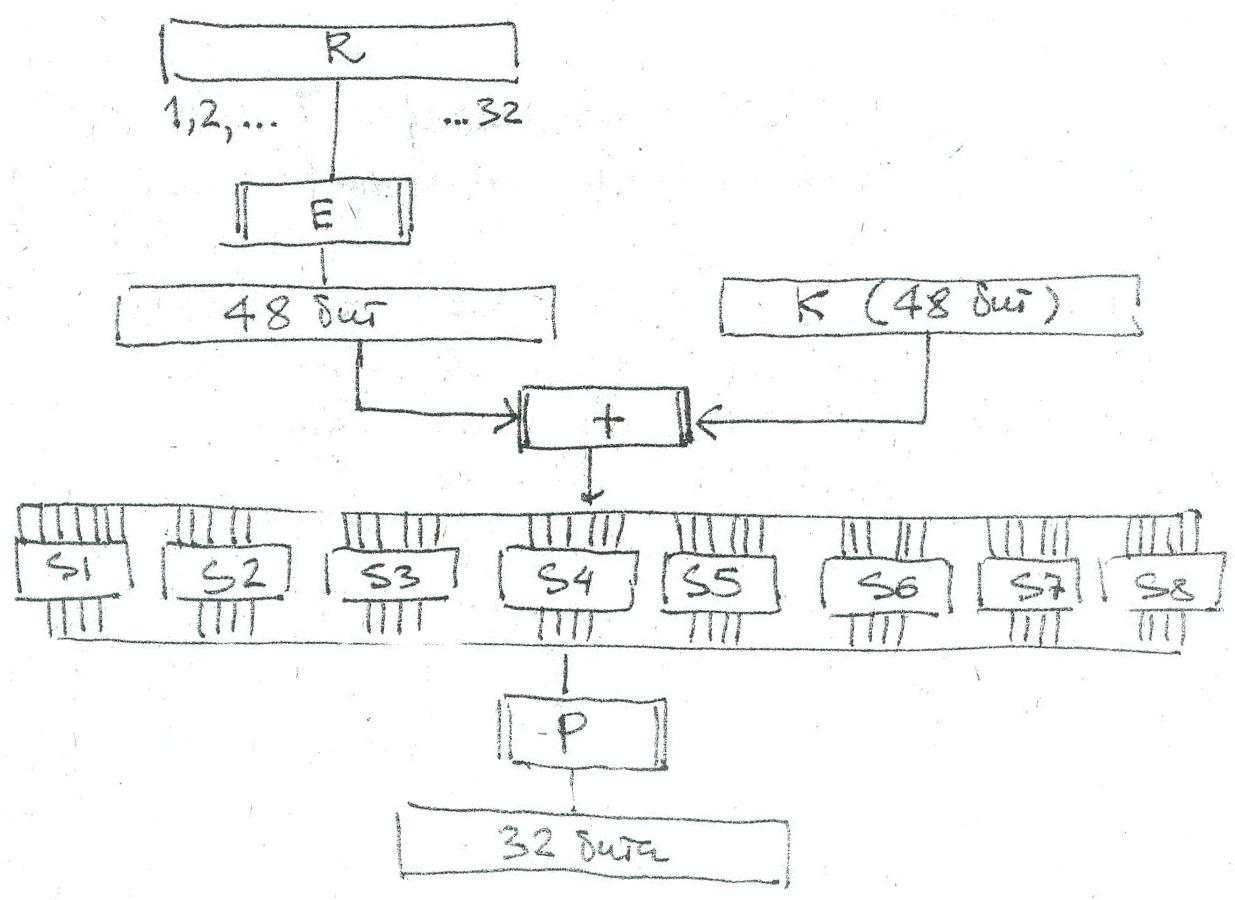
IP^(-1) - Конечная перестановка

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

#### Функция Фейстеля

Алгоритм:

1. Е - расширяющая перестановка R
2. xor с K(i)
3. S-замены
4. Конечная перестановка P



E – функция расширения 32 бит до 48 бит (дублирование некоторых бит)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

Si – преобразование 6 бит в 4:

* 0+5 – номер строки
* 1+2+3+4 – номер столбца

Р – перестановка на завершающем шаге функции шифрования F

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

В последнем раунде нет R<->L, чтобы алгоритм был симметричным.

8 s-блоков

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S0 | 0 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 1 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 2 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 3 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S1 | 0 | 15 | 1 | 8 | 14 | 6 | 11 | 3 | 4 | 9 | 7 | 2 | 13 | 12 | 0 | 5 | 10 |
| 1 | 3 | 13 | 4 | 7 | 15 | 2 | 8 | 14 | 12 | 0 | 1 | 10 | 6 | 9 | 11 | 5 |
| 2 | 0 | 14 | 7 | 11 | 10 | 4 | 13 | 1 | 5 | 8 | 12 | 6 | 9 | 3 | 2 | 15 |
| 3 | 13 | 8 | 10 | 1 | 3 | 15 | 4 | 2 | 11 | 6 | 7 | 12 | 0 | 5 | 14 | 9 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S2 | 0 | 10 | 0 | 9 | 14 | 6 | 3 | 15 | 5 | 1 | 13 | 12 | 7 | 11 | 4 | 2 | 8 |
| 1 | 13 | 7 | 0 | 9 | 3 | 4 | 6 | 10 | 2 | 8 | 5 | 14 | 12 | 11 | 15 | 1 |
| 2 | 13 | 6 | 4 | 9 | 8 | 15 | 3 | 0 | 11 | 1 | 2 | 12 | 5 | 10 | 14 | 7 |
| 3 | 1 | 10 | 13 | 0 | 6 | 9 | 8 | 7 | 4 | 15 | 14 | 3 | 11 | 5 | 2 | 12 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S3 | 0 | 7 | 13 | 14 | 3 | 0 | 6 | 9 | 10 | 1 | 2 | 8 | 5 | 11 | 12 | 4 | 15 |
| 1 | 13 | 8 | 11 | 5 | 6 | 15 | 0 | 3 | 4 | 7 | 2 | 12 | 1 | 10 | 14 | 9 |
| 2 | 10 | 6 | 9 | 0 | 12 | 11 | 7 | 13 | 15 | 1 | 3 | 14 | 5 | 2 | 8 | 4 |
| 3 | 3 | 15 | 0 | 6 | 10 | 1 | 13 | 8 | 9 | 4 | 5 | 11 | 12 | 7 | 2 | 14 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S4 | 0 | 2 | 12 | 4 | 1 | 7 | 10 | 11 | 6 | 8 | 5 | 3 | 15 | 13 | 0 | 14 | 9 |
| 1 | 14 | 11 | 2 | 12 | 4 | 7 | 13 | 1 | 5 | 0 | 15 | 10 | 3 | 9 | 8 | 6 |
| 2 | 4 | 2 | 1 | 11 | 10 | 13 | 7 | 8 | 15 | 9 | 12 | 5 | 6 | 3 | 0 | 14 |
| 3 | 11 | 8 | 12 | 7 | 1 | 14 | 2 | 13 | 6 | 15 | 0 | 9 | 10 | 4 | 5 | 3 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S5 | 0 | 12 | 1 | 10 | 15 | 9 | 2 | 6 | 8 | 0 | 13 | 3 | 4 | 14 | 7 | 5 | 11 |
| 1 | 10 | 15 | 4 | 2 | 7 | 12 | 9 | 5 | 6 | 1 | 13 | 14 | 0 | 11 | 3 | 8 |
| 2 | 9 | 14 | 15 | 5 | 2 | 8 | 12 | 3 | 7 | 0 | 4 | 10 | 1 | 13 | 11 | 6 |
| 3 | 4 | 3 | 2 | 12 | 9 | 5 | 15 | 10 | 11 | 14 | 1 | 7 | 6 | 0 | 8 | 13 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S6 | 0 | 4 | 11 | 2 | 14 | 15 | 0 | 8 | 13 | 3 | 12 | 9 | 7 | 5 | 10 | 6 | 1 |
| 1 | 13 | 0 | 11 | 7 | 4 | 9 | 1 | 10 | 14 | 3 | 5 | 12 | 2 | 15 | 8 | 6 |
| 2 | 1 | 4 | 11 | 13 | 12 | 3 | 7 | 14 | 10 | 15 | 6 | 8 | 0 | 5 | 9 | 2 |
| 3 | 6 | 11 | 13 | 8 | 1 | 4 | 10 | 7 | 9 | 5 | 0 | 15 | 14 | 2 | 3 | 12 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| S7 | 0 | 13 | 2 | 8 | 4 | 6 | 15 | 11 | 1 | 10 | 9 | 3 | 14 | 5 | 0 | 12 | 7 |
| 1 | 1 | 15 | 13 | 8 | 10 | 3 | 7 | 4 | 12 | 5 | 6 | 11 | 0 | 14 | 9 | 2 |
| 2 | 7 | 11 | 4 | 1 | 9 | 12 | 14 | 2 | 0 | 6 | 10 | 13 | 15 | 3 | 5 | 8 |
| 3 | 2 | 1 | 14 | 7 | 4 | 10 | 8 | 13 | 15 | 12 | 9 | 0 | 3 | 5 | 6 | 11 |

-------------------------------------------------------45 минут---------------------------------------------

#### Расшифровка текста

Выполняется в обратном порядке:

1. Перестановка по таблице начальной перестановки (так как она взаимно обратна конечной)
2. На каждом из 16 раундов:

R(i-1)=L(i)

L(i-1)=R(i) +F(R(i), K(i))

1. На последнем шаге получаем L(0)+R(0) и переставляем в соответствии с таблицей конечной перестановки

### Взлом шифра DES:

Основной недостаток DES – короткий ключ (можно взломать перебором всех возможных ключей)

1. 2002 год – взлом за 96 дней
2. 2004 год – взлом за 22 часа с помощью 100 000 компьютеров

### Усовершенствования DES

Используют тройное шифрование DES по ключам K1, K2 и K3, что увеличивает количество операций до 10^34

С=Еk3(Dk2(Ek1(P)))

P=Dk1(Ek2(Dk3(C)))

Дает фактическое увеличение длины ключа до 192 бит

## Advanced Encryption Standard (AES)

1997 - объявлен открытый конкурс - National Institute of Standards and Technology (NIST)

Первый тур – 15 алгоритмов, второй – 5

2 октября 2000 года - объявлен победитель конкурса, проведенного NIST – Rijndael

Разработан Джоном Дэменом и Винсентом Риджменом из **Католического университета города Лювена, Бельгия**.

26 ноября 2001 года - опубликована окончательная спецификация AES

2002 - правительство США объявило **стандартом шифрования**, сертифицирован Агентством национальной безопасности США (National Security Agency/Central Security Service, NSA/CSS).

NIST: компьютер, взламывающий DES за секунду, ломал бы AES с 128-битным ключом 149 триллионов лет (7 возрастов Вселенной).

### Требования к алгоритмам

Обязательные:

1. Блок данных: 128 бит
2. Размеры ключа: 128, 192 и 256 бит

Рекомендательные:

1. Алгоритм должен быть **стойким против криптоаналитических атак**, известных на время проведения конкурса.
2. **Ясная, простая и обоснованная структура алгоритма**.
   1. облегчение **анализ алгоритма** в рамках конкурса
   2. гарантия **отсутствия в алгоритме «закладок»**, специально внедренных авторами.
3. Должны **отсутствовать слабые и эквивалентные ключи** (т.е. ключи, являющиеся различными, но приводящие к одному и тому же результату шифрования).
4. **Скорость шифрования данных высокая** на всех потенциальных аппаратных платформах – от 8-битных до 64-битных.
5. Структура алгоритма должна позволять **распараллеливание операций** в многопроцессорных системах и аппаратных реализациях.
6. Алгоритм должен предъявлять **минимальные требования к памяти** (оперативной и энергонезависимой).
7. Не должно быть **ограничений для использования алгоритма** в качестве основы для:
   1. построения хэш-функций,
   2. генераторов псевдослучайных последовательностей

Недостатки алгоритмов

MARS –

RC6 –

Rijndael –

Serpent –

Twofish –

### Параметры алгоритма

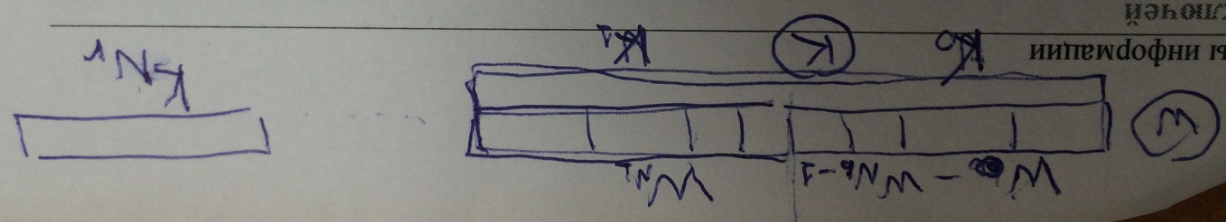
Длина блока входных данных i (как и длина текущего состояния, обозначаемого s) постоянна и равна **128 битам**. Длина ключа k составляет 128, 192 либо 256 бит.

Для обозначения длины используется кратность 32 битам: длина входного блока Nb=4, а длина ключа Nk=4, 6 или 8.

L(Mi)=L(S)=128b=16 байт=4 слова=Nb

|  |  |  |
| --- | --- | --- |
| **Размер ключа Nk (\*32)** | **Количество раундов Nr** | **Количество возможных комбинаций** |
| 128 (4) | 10 | 340\*10^36 |
| 192 (6) | 12 |  |
| 256 (8) | 14 | 11\*10^76 |

### Подготовка ключей



Из исходного ключа k формируется массив w, длина которого равна **Nb\*(Nr+1)**, где Nr – количество итераций (называемых раундами), равное 10, 12 или 14 раз (зависит от длины ключа - Nk).

1. Первые Nk 32 битных слов массива w заполняются исходным ключом k.
2. Остальные слова (Nk+1 и дальше)

* **i кратно Nk**:

Каждое слово w[i] определяется как результат операции XOR для S-перестановки 32 битного слова, полученного циклическим сдвигом на один байт влево предыдущего слова w[i-1] ключа, и массива Rcon(x), содержащего значения (2^(x-1) mod 256, 0, 0, 0) в поле Галуа (GF(2^8)). После этого выполняется XOR со словом ключа предыдущего раунда (w[i-Nk]).

*w[i]={SubBytes(ShiftRow(w[i-1]))* ***XOR*** *Rcon(i/Nk)}* ***XOR*** *w[i-Nk]*

Алгоритм:

* i mod Nk=0 (первый блок очередного раундового ключа):

1. Wi-1
   1. ShiftRow
   2. SubBytes
2. xor RCon
3. xor Wi-Nk

* **Для 256 битного ключа** (Nk=8), для i-х слов, если i(mod(Nk))=4, выполняется только S-подстановка w[i-1]

*w[i]={SubBytes (w[i-1]) } XOR w[i-Nk]*

Алгоритм:

Nk=8 и i(mod(Nk))=4

1. SubBytes (Wi-1)
2. xor Wi-Nk

* Остальные i - XOR со словом ключа предыдущего раунда (w[i-Nk]).

*w[i]= w[i-1] XOR w[i-Nk]*

Ключ для итерации i входит в массив w в ячейках [Nb\*i, Nb\*(i+1)].

### Вычисление RCon

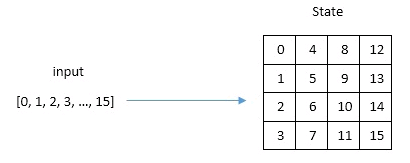
Массив RCon(x) содержит значения (2^(x-1) mod (0х11B), 0, 0, 0).

Предыдущее значение первого байта умножается на 2. В частности, 0х80\*2=0х100=>0x100 xor 0x11B=0x1B

Сокращение по модулю 0х11B (1 0001 1011b) выполняется только, если значение превышает 0xFF

### Шифрование сообщения

Для шифрования сообщения m оно разбивается на блоки m(i), каждый из которых шифруется отдельно. Выделенный блок m(i) копируется в массив текущего состояния s по следующей формуле: s[r, c]=m(i)[r+4\*c], для r=[0,4) и c=[0, Nb).



Алгоритм:

1. Расширить ключ (получить массив ключей W)
2. xor c K0
3. *SubBytes - подстановка s-box*
4. *ShiftRows - shl (строк)*
5. *кроме последней: MixColumns - с(х)*
6. *AddRoundKey - xor Ki*
7. *Повторить Nr раз с п.3*



3 операции повторяются на каждой итерации (раунде):

* **нелинейное замещение байтов массива s по таблице S-подстановок**. Для каждого байта a(i) массива s получаем байт b(i).

Процедура SubBytes выполняет табличную s-подстановку, используя таблицу hex-чисел (байт):

Si=0xXY -> Si’=[X,Y]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 76 |
| CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 53 | D1 | 0 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | 3C | 9F | A8 |
| 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| E0 | 32 | 3A | 0A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| 70 | 3E | B5 | 66 | 48 | 3 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

То есть входное значение 0х00 меняется на 0x63, 0х01 – на 0x7C и т.д.

Эта процедура эквивалентна вычислению мультипликативно обратной величины в конечном поле Галуа (0x00->0x00), с добавлением 0x63.

Вычисление мультипликативно обратной величины в 2 шага:

1. берется обратное число в конечном поле (поле Галуа) GF(2^8),
2. **каждый байт** b(i) побитово пересчитывается по формуле:

b(i)[j]= (b(i)[j]XOR(b(i)[j+4])mod(8)XOR(b(i)[j+5]))mod(8))XOR(b(i)[j+6])mod(8))XOR(b(i)[j+7]))mod(8))XOR(c[j]), для j=[0,8), и c=195(10)=1100 0011(2)

* Полученный массив подвергается **построчным циклическим сдвигам влево** по следующей формуле: b[i,j]=a[(i-j)mod(4),j] для i=[0,4) и j=[0, Nb).



* **перемножение каждой колонки массива s с многочленом c(x)**. Данная операция повторяется на всех раундах, кроме последнего. Перед перемножением четыре байта каждой колонки смешиваются, используя обратимую линейную перестановку. После этого каждая колонка представляется в виде полинома четвертой степени, который умножается на многочлен c(x)=3x^3+x^2+x+2 поле GF(2^8) по модулю x^4+1.



2 шага:

1. умножение полиномов
2. деление с остатком

Пример: 5 (101)\*7 (111)=10

5\*7=>(x^2+1)(x^2+x+1)=x^4+x^2+ x^3+x+ x^2+1=x^4+ x^3+x+1

=>(x^4+ x^3+x+1)/(x^4+1)=1 с остатком (x^3+x)=>1010=10

* **побитовый XOR раундового ключа** k(i) с s.

Полученная на последнем раунде матрица состояния S копируется в выходное (зашифрованное) сообщение (строки склеиваются одна за другой) по формуле сm(i)[r+4\*c]= s[r, c], для r=[0,4) и c=[0, Nb).

Перед первым раундом выполняется **побитовый XOR раундового ключа** k(0) с s, а на последнем раунде не выполняется **перемножение каждой колонки массива s с многочленом.**

### Расшифровка сообщения



После этого выполняется **перемножение каждой колонки массива s с многочленом c(x)**. Данная операция повторяется на всех раундах, кроме последнего. Перед перемножением четыре байта каждой колонки смешиваются, используя обратимую линейную перестановку. После этого каждая колонка представляется в виде полинома четвертой степени, который умножается на многочлен c(x)^(-1)=0B\*x^3+0D\*x^2+09\*x+0E поле GF(2^8) по модулю x^4+1.



### Применение

Поддержка АЕS (и только его) реализована Intel на всех процессорах, выпущенных с 2009 года

(начиная с версии Core i7-980X Extreme Edition (год?), а затем в процессорах Sandy Bridge). Сейчас внедрена в каждом i5.

Для этого реализованы специальные инструкции:

AESENC (AESENCLAST)

AESDEC (AESDECLAST)