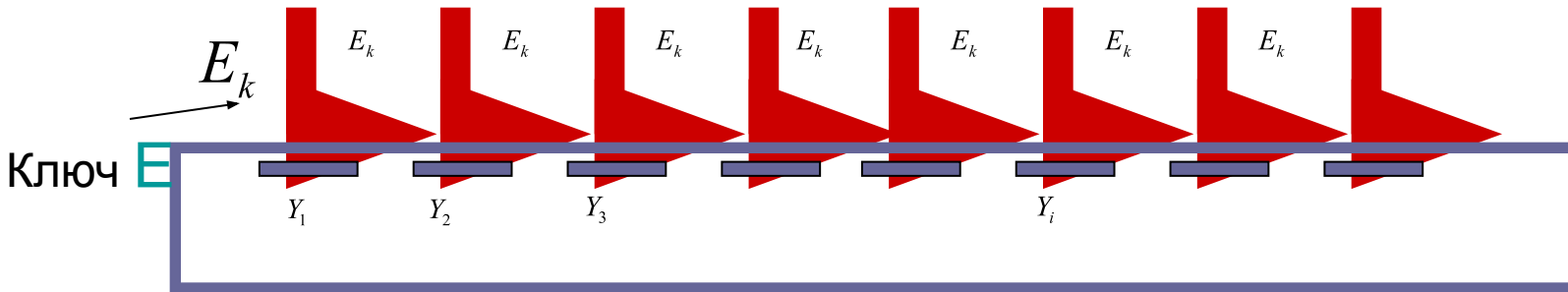
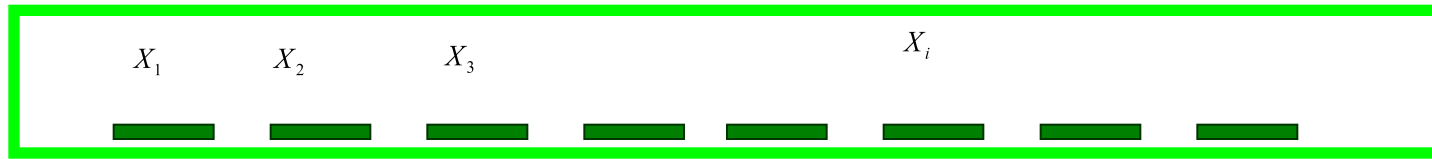


# Симметричные блочные шифры

Открытый текст = Plane text



Шифрованный текст

$$Y_i = E_k(X_i), i = 1, 2, \dots - \text{шифрование}$$

$$X_i = E_k^{-1}(Y_i), i = 1, 2, \dots - \text{дешифрование}$$

Блочными называются шифры, в которых логической единицей шифрования является некоторый блок открытого текста, после преобразования которого получается блок шифрованного текста такой же длины. Обычно используются блоки длиной 64 бита (128). Большинство сетевых приложений, в которых применяются схемы симметричного шифрования, используют именно блочные шифры.

Пусть блочный шифр оперирует с блоками длины  $n$ . Всего таких блоков  $2^n$ . Обратимых преобразований блоков длины  $n$  в блоки такого же размера всего  $(2^n)!$ . Если  $n$  невелико, то такой шифр эквивалентен шифру подстановки на соответствующем алфавите и нестойк к частотному анализу. Чем больше  $n$ , тем менее эффективна статистическая атака. Выписать таблицу преобразования блоков длины 64 бита не представляется возможным.

# DES

## История

В 1972 году было проведено исследование потребности правительства США в компьютерной безопасности. Американское «национальное бюро стандартов» (НБС) (ныне известное, как NIST — «национальный институт стандартов и технологий») определило необходимость в общеправительственном стандарте шифрования некритичной информации.

НБС проконсультировалось с АНБ (агентством национальной безопасности США) и 15 мая 1973 года объявило первый конкурс на создание шифра. Были сформулированы строгие требования к новому шифру. Фирма IBM представила на конкурсе разработанный её шифр, называемый «Люцифер» (Lucifer). Шифры ни одного из конкурсантов (включая «Люцифер») не обеспечивали выполнение всех требований. В течение 1973-1974 годов IBM доработала свой «Люцифер»: использовала в его основе алгоритм Хорста Фейстеля, созданный ранее. 27 августа 1974 года начался второй конкурс. На сей раз шифр «Люцифер» сочли приемлемым.

17 марта 1975 года предложенный алгоритм DES был издан в «Федеральном реестре». В 1976 году для обсуждения DES было проведено два открытых симпозиума. На симпозиумах жёсткой критике подверглись изменения, внесённые в алгоритм организацией АНБ. АНБ уменьшило первоначальную длину ключа и S-блоки (блоки подстановки), критерии проектирования которых не раскрывались. АНБ подозревалось в сознательном ослаблении алгоритма с той целью, чтобы АНБ могло легко просматривать зашифрованные сообщения. Сенат США проверил действия АНБ и в 1978 году опубликовал заявление, в котором сообщалось следующее:

в процессе разработки алгоритма представители АНБ убедили создателей DES в том, что уменьшенной длины ключа более чем достаточно для всех коммерческих приложений;

представители АНБ косвенно помогали в разработке S-перестановок;

окончательная версия алгоритма была, по мнению проверяющих, лучшим алгоритмом шифрования, к тому же лишённым статистической или математической слабости;

представители АНБ никогда не вмешивались в разработку алгоритма DES.

В 1990 году Эли Бихам (Eli Biham) и Ади Шамир (Adi Shamir) провели независимые исследования по дифференциальному криптоанализу — основному методу взлома блочных алгоритмов симметричного шифрования. Эти исследования сняли часть подозрений в скрытой слабости S-перестановок. S-блоки алгоритма DES оказались намного более устойчивыми к атакам, чем если бы их выбрали случайно.

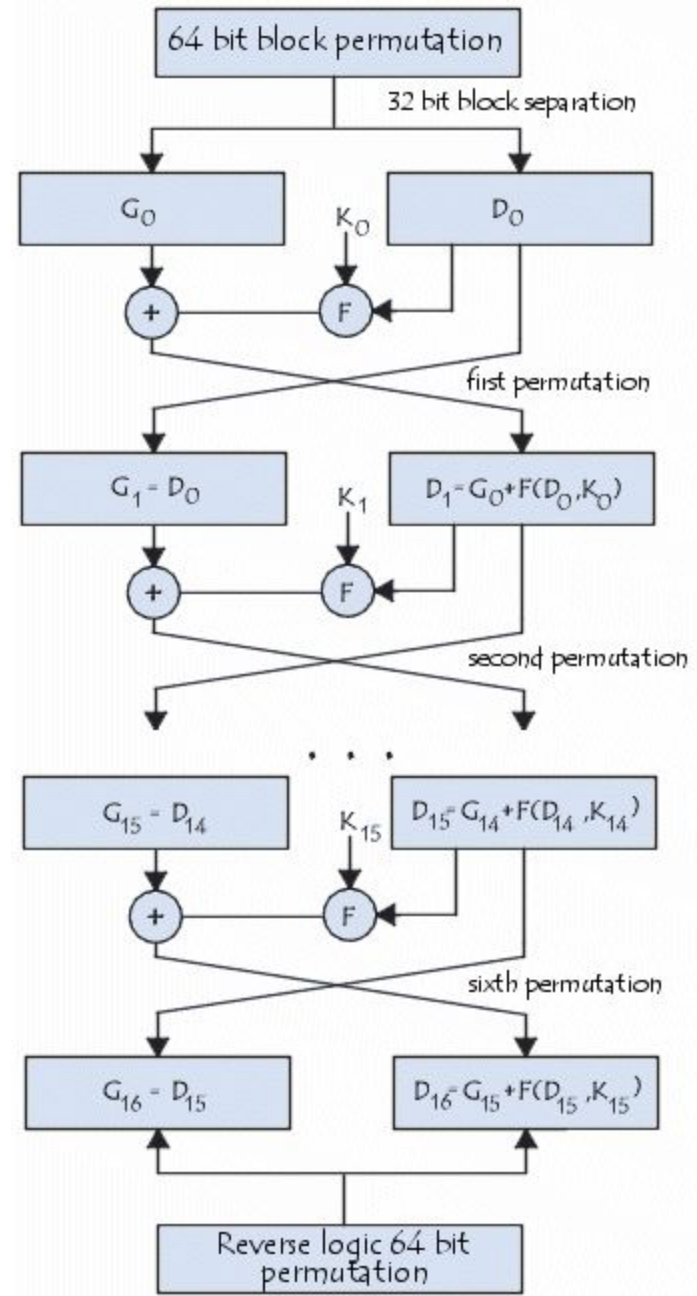
Алгоритм DES – шифр Фейстеля со следующими параметрами

- 1) 64 – битные блоки;
- 2) 56 битный ключ;
- 3) 16 раундов шифрования
- 4) Функция  $F$  –композиционная.

Схема шифрования.

Открытый текст  $\rightarrow$  Начальная перестановка  $IP \rightarrow$  Шифр Фейстеля  $\rightarrow IP^{-1} \rightarrow$  Шифрованный текст

В основе алгоритма лежит цепь Фейстеля, чья раундовая функция  $F$  обрабатывает 32-разрядные слова (половину блока) и использует в качестве параметра 48-разрядный подключ



## Цикловые ключи.

### 1) Начальная перестановка с выбором PC1.

Используется 64-битный ключ.

Сначала к нему применяется функция перестановки с выбором PC1.

Получается 56 битное значение.

PC1=

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

### 2) Циклический сдвиг.

Полученное 56-битовое значение рассматривается как комбинация двух 28-битовых значений. В каждом раунде шифрования к правой и левой части ключа по отдельности применяются операции циклического сдвига влево на 1 или 2 бита в соответствии с таблицей сдвигов.

Таблица сдвигов

1	1	2	2	2	2	2	2	1	2	2	2	2	2	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

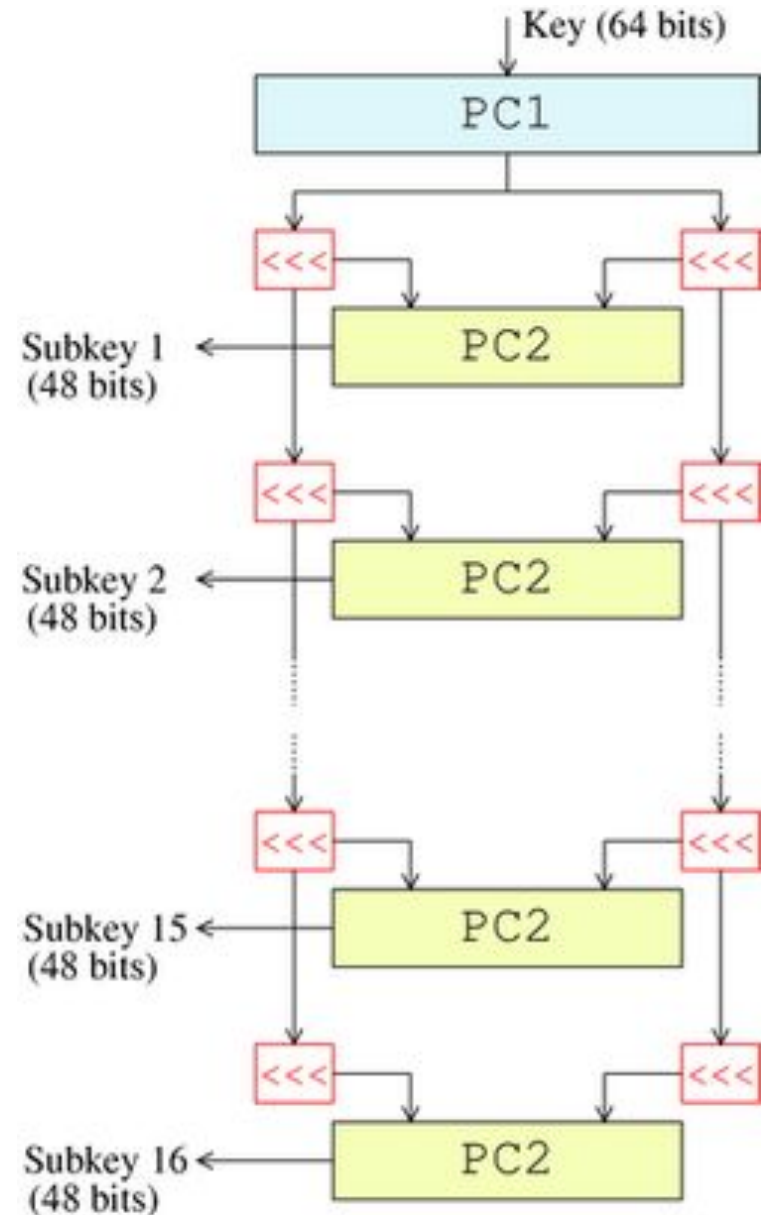
### 3) Перестановка PC2

Затем с помощью другой перестановки PC2 из полученного результата для каждого из 16 раундов генерируется подключ K<sub>i</sub> длины 48 битов. Функция перестановки одна и та же для всех раундов, но генерируемые подключи оказываются разными из-за того, что в результате циклического сдвига на ее вход поступают разные биты ключа.

PC2					
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

1-ый бит отображается в 14 позицию

2-ой бит отображается в 17 позицию



## 2. Циклический сдвиг.

Полученное 56-битовое значение рассматривается как комбинация двух 28-битовых значений. В каждом раунде шифрования к правой и левой части ключа по отдельности применяются операции циклического сдвига влево на 1 или 2 бита в соответствии с таблицей сдвигов.

**Таблица сдвигов**

1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

DES also has four so-called [weak keys](#). Encryption ( $E$ ) and decryption ( $D$ ) under a weak key have the same effect (see [involution](#)):

or equivalently,

There are also six pairs of *semi-weak keys*. Encryption with one of the pair of semiweak keys,  $K_1$ , operates identically to decryption with the other,  $K_2$ :

or equivalently,

It is easy enough to avoid the weak and semiweak keys in an implementation, either by testing for them explicitly, or simply by choosing keys randomly; the odd

DES has also been proved not to be a [group](#), or more precisely, the set  $\{E_K\}$  (for all possible keys  $K$ ) under [functional composition](#) is not a group, nor "close" to being

It is known that the maximum cryptographic security of DES is limited to about 64 bits, even when independently choosing all round subkeys instead of deriving

$$E_K(E_K(P)) = P = D_K(P).$$

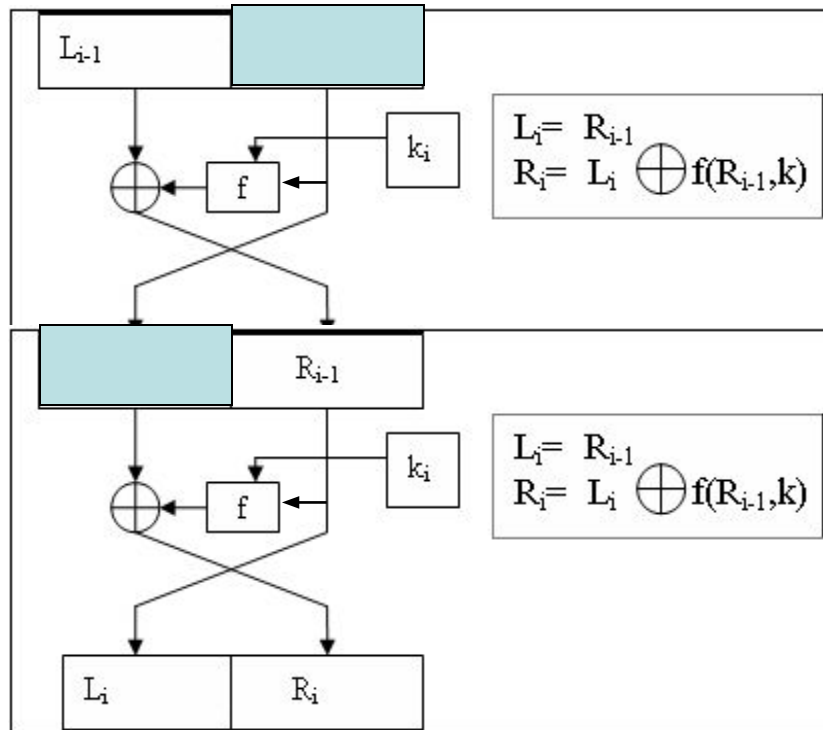
$$E_{K_1}(E_{K_2}(P)) = P = D_{K_1}(E_{K_2}(P)).$$

$K_1$

$K_2$

# Прямое и обратное преобразование сети Фейстеля

# Криптостойкость и диффузия



1	4
2	16
3	64
4	256
5	1024
6	4096
7	16384
8	65536

Нарастание диффузии от раунда к раунду.

Стойкость шифра часто измеряют в относительном числе раундов, которые поддаются взлому.



Кроме того, практически сразу после появления DES были обнаружены следующие проблемы с ключами шифрования DES [4]:

1. 4 ключа из возможных 256 ключей алгоритма являются слабыми (т. е. не обеспечивают требуемой стойкости при зашифровании). Это ключи, в которых все биты какой-либо из половин расширяемого ключа (С или D на рис. 1) являются нулевыми или единичными. В этом случае все ключи раунда будут одинаковыми.

2. 6 пар ключей являются эквивалентными (т. е. информация, зашифрованная одним ключом из пары, расшифровывается другим ключом той же пары), например, пара ключей E0FEE0FEF1FEF1FE165 и FEE0FEE0FEE1FEE116. Процедура расширения такого ключа вместо 16 различных ключей раунда вырабатывает всего 2.

3. 48 ключей являются «возможно слабыми», их полный список приведен в [4]. Возможно слабые ключи при их расширении дают только 4 различных ключа раунда, каждый из которых используется при шифровании по 4 раза.

4. Существует свойство комплементарности ключей: если

$$E_k(M) = C,$$

где  $E_k(M)$  – зашифрование алгоритмом DES блока открытого текста  $M$ , а  $C$  – полученный в результате шифртекст, то

$$E_{k'}(M') = C',$$

где  $x'$  – побитовое дополнение к  $x$  (т. е. величина, полученная путем замены всех битовых нулей значения  $x$  на единицы, и наоборот).

## Отечественный стандарт шифрования ГОСТ 28147-89

«ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования»

[ГОСТ Р 34.12-2015](#),

[ГОСТ 34.12-2018](#)

ГОСТ 28147-89 - это стандарт, принятый в 1989 году в Советском Союзе и установивший алгоритм шифрования данных, составляющих гостайну. По свидетельству причастных к его реализациям и использованию людей, алгоритм был разработан в 70-е годы в 8-м Главном Управлении КГБ СССР, тогда он имел гриф Сов.Секретно. Затем гриф был понижен до Секретно, а когда в 89-м году алгоритм был проведен через Госстандарт и стал официальным государственным стандартом, гриф с него был снят. В начале 90-х годов он стал полностью открытым.

Стандарты СНГ ([ГОСТ](#))

[ГОСТ 28147-89](#). Системы обработки информации. Защита криптографическая.

Алгоритм криптографического преобразования

[ГОСТ 34.10-2018](#). Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи

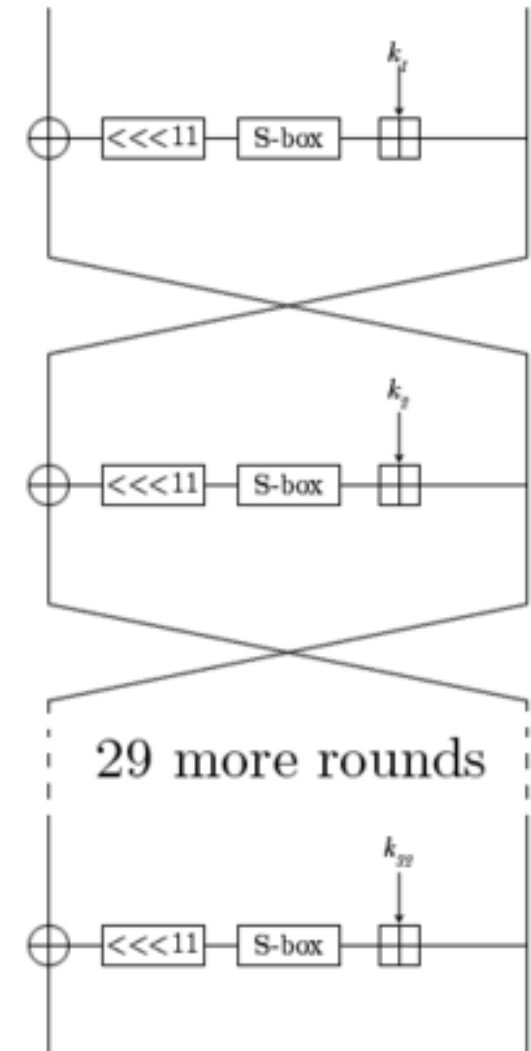
[ГОСТ 34.11-2018](#). Информационная технология. Криптографическая защита информации. Функция хеширования

[ГОСТ 34.12-2018](#). Информационная технология. Криптографическая защита информации. Блочные шифры

[ГОСТ 34.13-2018](#). Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров

- ГОСТ 28147-89
- -Шифр Фейстеля.

- 1) размер шифруемого блока равен 64 битам;
- 2) размер ключа равен 256 битам;
- 3) число шагов цикла  $n = 32$ ;
- 4) S –блоки не фиксированы.



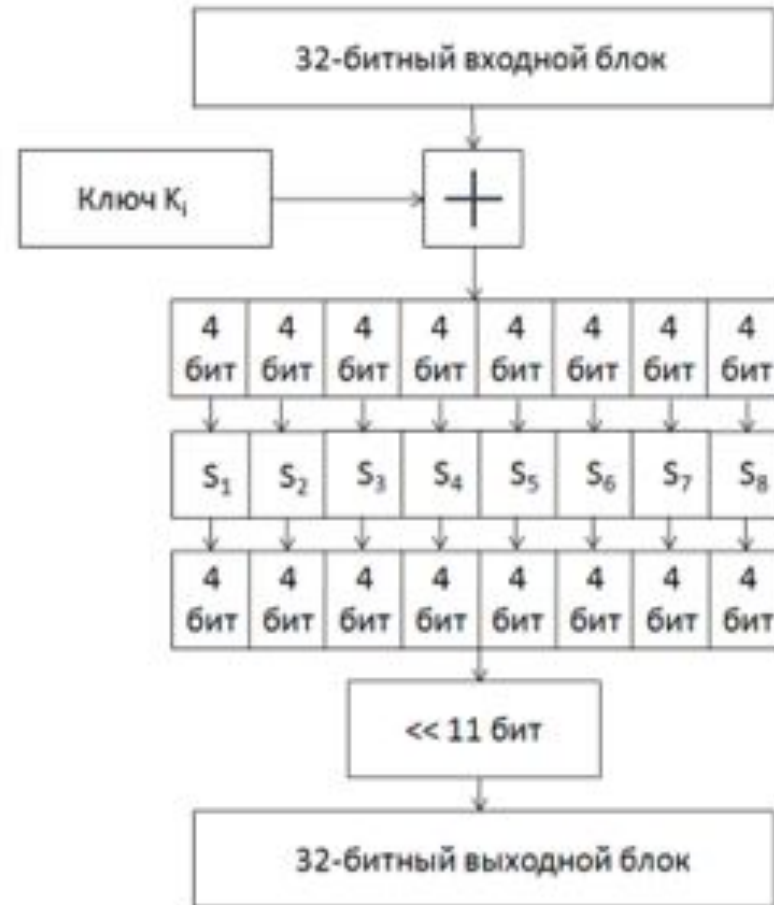
- 4) получение цикловых ключей
- Ключ  $K$  длины 256 битов разбивается на блоки длиной 32 бит:
- $K = P_1 P_2 \dots P_8$  ,  $|P_i| = 32, i = 1, \dots, 8.$
- При шифровании используется ключевая последовательность:
- $(K_1, K_2, \dots, K_{31}, K_{32}) = (P_1, P_2, \dots, P_8, P_1, P_2, \dots, P_8, P_1, P_2, \dots, P_8, P_8, P_7, \dots, P_1),$

# Функция усложнения F:

$$w_1 = r \oplus k \bmod 2^{32}$$

$$w_2 = f_s(w_1)$$

$$w_3 = T^{11}(w_2)$$



**Таблица замен в ГОСТе - аналог S-блоков DES'a - представляет собой таблицу (матрицу) размером 8x16, содержащую число от 0 до 15. В каждой строке каждое из 16-ти чисел должно встретиться ровно 1 раз. Таблица замен в ГОСТе одна и та же для всех раундов и, (В отличие от DES'a, ) не зафиксирована в стандарте. От качества этой таблицы зависит качество шифра. При "сильной" таблице замен стойкость шифра не опускается ниже некоторого допустимого предела даже в случае ее разглашения. И наоборот, использование "слабой" таблицы может уменьшить стойкость шифра до недопустимо низкого предела. Никакой информации по качеству таблицы замен в открытой печати России не публиковалось, однако существование "слабых" таблиц не вызывает сомнения - примером может служить "тривиальная" таблица замен, по которой каждое значение заменяется на него самого. Это делает ненужным для компетентных органов России ограничивать длину ключа - можно просто поставить недостаточно "сильную" таблицу замен.**

**В настоящее время известны *S-boxes*, которые используются в приложениях Центрального Банка РФ и считаются достаточно сильными. Напомню, что входом и выходом *S-box* являются 4-битные числа, поэтому каждый *S-box* может быть представлен в виде строки чисел от 0 до 15, расположенных в некотором порядке. Тогда порядковый номер числа будет являться входным значением *S-box*, а само число - выходным значением *S-box*.**

0123456789ABCDEF  
1 4A92D80E6B1C7F53  
2 EB4C6DFA23810759  
3 581DA342EFC7609B  
4 7DA1089FE46CB253  
5 6C715FD84A9E03B2  
6 4BA0721D36859CFE  
7 DB413F590AE7682C  
8 1FD057A4923E6B8C

Данный набор *S*-блоков используется в криптографических приложениях ЦБ РФ.

## Криптоанализ

В мае 2011 года известный криптоаналитик [Николя Куртуа](#) заявил об обнаружении серьезных уязвимостей в данном шифре.

## Критика ГОСТа

Основные проблемы ГОСТа связаны с неполнотой стандарта в части генерации ключей и таблиц замен. Тривиально доказывается, что у ГОСТа существуют «слабые» ключи и таблицы замен, но в стандарте не описываются критерии выбора и отсева «слабых». Также стандарт не специфицирует алгоритм генерации таблицы замен (S-блоков). С одной стороны, это может являться дополнительной секретной информацией (помимо ключа), а с другой, поднимает ряд проблем:

нельзя определить криптостойкость алгоритма, не зная заранее таблицы замен;

реализации алгоритма от различных производителей могут использовать разные таблицы замен и могут быть несовместимы между собой;

возможность преднамеренного предоставления слабых таблиц замен лицензирующими органами РФ;

потенциальная возможность (отсутствие запрета в стандарте) использования таблиц замены, в которых узлы не являются перестановками, что может привести к чрезвычайному снижению стойкости шифра.



# Сравнение ГОСТ и DES

ПАРАМЕТР	ГОСТ	DES
1. Размер блока шифрования	64 бита	64 бита
2. Длина ключа	256 бит	56 бит
3. Число раундов	32	16
4. Узлы замен (S-блоки)	не фиксированы	фиксированы
5. Длина ключа для одного раунда	32 бита	48 бит
6. Схема выработки раундового ключа	простая	сложная
7. Начальная и конечная перестановки битов	нет	есть

в DES сложная генерация подключей из ключей, а в ГОСТ простая.  
в DES 56-битный ключ, а ГОСТ 256-битный + секретные S-блоки.  
в DES 16 раундов, а в ГОСТ - 32 раунда.  
в DES используются нерегулярные перестановки (P-блоки), а в ГОСТ 11-битный циклический сдвиг.

**DES** использует гораздо более сложную процедуру создания *подключей*, чем *ГОСТ 28147*. В *ГОСТ* эта процедура очень проста. В **DES** применяется 56-битный ключ, а в *ГОСТ 28147* - 256-битный. При выборе сильных *S-boxes* *ГОСТ 28147* считается очень стойким. У *S-boxes* **DES** 6-битные входы и 4-битные выходы, а у *S-boxes* *ГОСТ 28147* 4-битные входы и выходы. В обоих алгоритмах используется по восемь *S-boxes*, но размер *S-box* *ГОСТ 28147* существенно меньше размера *S-box* **DES**.

В **DES** применяются нерегулярные перестановки *P*, в *ГОСТ 28147* используется 11-битный циклический сдвиг влево. Перестановка **DES** увеличивает лавинный эффект. В *ГОСТ 28147* изменение одного входного бита влияет на один *S-box* одного раунда, который затем влияет на два *S-boxes* следующего раунда, три *S-boxes* следующего и т. д. В *ГОСТ 28147* требуется 8 раундов прежде, чем изменение одного входного бита повлияет на каждый бит результата; **DES** для этого нужно только 5 раундов.

В **DES** 16 раундов, в *ГОСТ 28147* - 32 раунда, что делает его более стойким к дифференциальному и линейному криптоанализу.

ГОСТ не запатентован, поэтому его может свободно использовать любое юридическое и физическое лицо, если, конечно, это не противоречит законодательству страны где находятся это лицо. Со стороны авторов ГОСТа претензий нет и быть не может, так как юридические права на алгоритм ни за кем не закреплены.

# Слабые ключи DES и ГОСТ

- Слабыми ключами называется ключи  $k$  такие, что
- $E_k E_k(x) = x$
- , где  $x$  — 64-битный блок.

## Алгоритм IDEA

### International Data Encryption Algorithm

*IDEA* является одним из нескольких симметричных криптографических алгоритмов, которыми первоначально предполагалось заменить DES.

Является блочным симметричным алгоритмом шифрования, разработанным Сюэзя Лай (Xuejia Lai) и Джеймсом Массей (James Massey) из швейцарского федерального института технологий. Первоначальная версия была опубликована в 1990 году.



## Алгоритм IDEA

### International Data Encryption Algorithm

*IDEA* является одним из нескольких симметричных криптографических алгоритмов, которыми первоначально предполагалось заменить DES.

Является блочным симметричным алгоритмом шифрования, разработанным Сюэця Лай (Xuejia Lai) и Джеймсом Массей (James Massey) из швейцарского федерального института технологий. Первоначальная версия была опубликована в 1990 году.

Структура шифра была разработана для легкого воплощения как программно, так и аппаратно.

**Длина блока:** длина блока должна быть достаточной, чтобы скрыть все статистические характеристики исходного сообщения. С другой стороны, сложность реализации криптографической функции возрастает экспоненциально в соответствии с размером блока. Использование блока размером в 64 бита в 90-е годы означало достаточную силу.

**Длина ключа:** длина ключа должна быть достаточно большой для того, чтобы предотвратить возможность простого перебора ключа. При длине ключа 128 бит *IDEA* считается достаточно безопасным.

**Конфузия:** зашифрованный текст должен зависеть от ключа сложным и запутанным способом.

**Диффузия:** каждый бит незашифрованного текста должен влиять на каждый бит зашифрованного текста. Распространение одного незашифрованного бита на большое количество зашифрованных бит скрывает статистическую структуру незашифрованного текста. Определить, как статистические характеристики зашифрованного текста зависят от статистических характеристик незашифрованного текста, должно быть непросто. *IDEA* с этой точки зрения является очень эффективным алгоритмом.

Частично основан на структуре FEAL

- 1) размер блока 64 бита;**
- 2) размер ключа 128 битов;**
- 3) число раундов 8:**
- 4) выходное отображение, обеспечивающее обратимость;**
- 5) входного отображения нет;**



Безопасность IDEA основывается на использовании трех не совместимых типов арифметических операций над 16-битными словами.

В ходе алгоритма выполняются следующие операции:

1. XOR сложение (+)
2. сложение по модулю  $2^{16}$  [+]
3. умножение по модулю  $2^{16} + 1$  (X) .

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

$$x \otimes (y [+ ] z) \neq x \otimes y [+ ] x \otimes z$$

$$x [+ ] (y \oplus z) \neq x [+ ] y \oplus x [+ ] z$$

$$x \otimes (y \oplus z) \neq x \otimes y \oplus x \otimes z$$

**Дистрибутивность** (от лат. *distributivus* — «распределительный») — свойство согласованности двух бинарных операций, определённых на одном и том же множестве.

# Ключевое расписание

Ключ К 128 бит.

Получаем 52 раундовых ключа – 6 на каждый их 8 раундов, 4- на выходное отображение;

Ключ К разбивается на 8 подблоков длины 16 бит.

Обозначаем

$K(1), K2(1), K3(1), K4(1), K5(1), K6(1), K1(2), K2(2)$ .

Затем Начальный ключ К циклически сдвигается влево на 25 бит и снова разбивается на 8 подблоков:

$K(2), K4(2), K5(2), K6(2), K1(3), K2(3), K3(3), K4(3)$ .

Затем снова сдвигается на 25 битов и снова разбивается, и так до тех пор, пока не сгенерируются 52 подблока.

1 цикл -  $K1(1), K2(1), K3(1), K4(1), K5(1), K6(1)$ ,

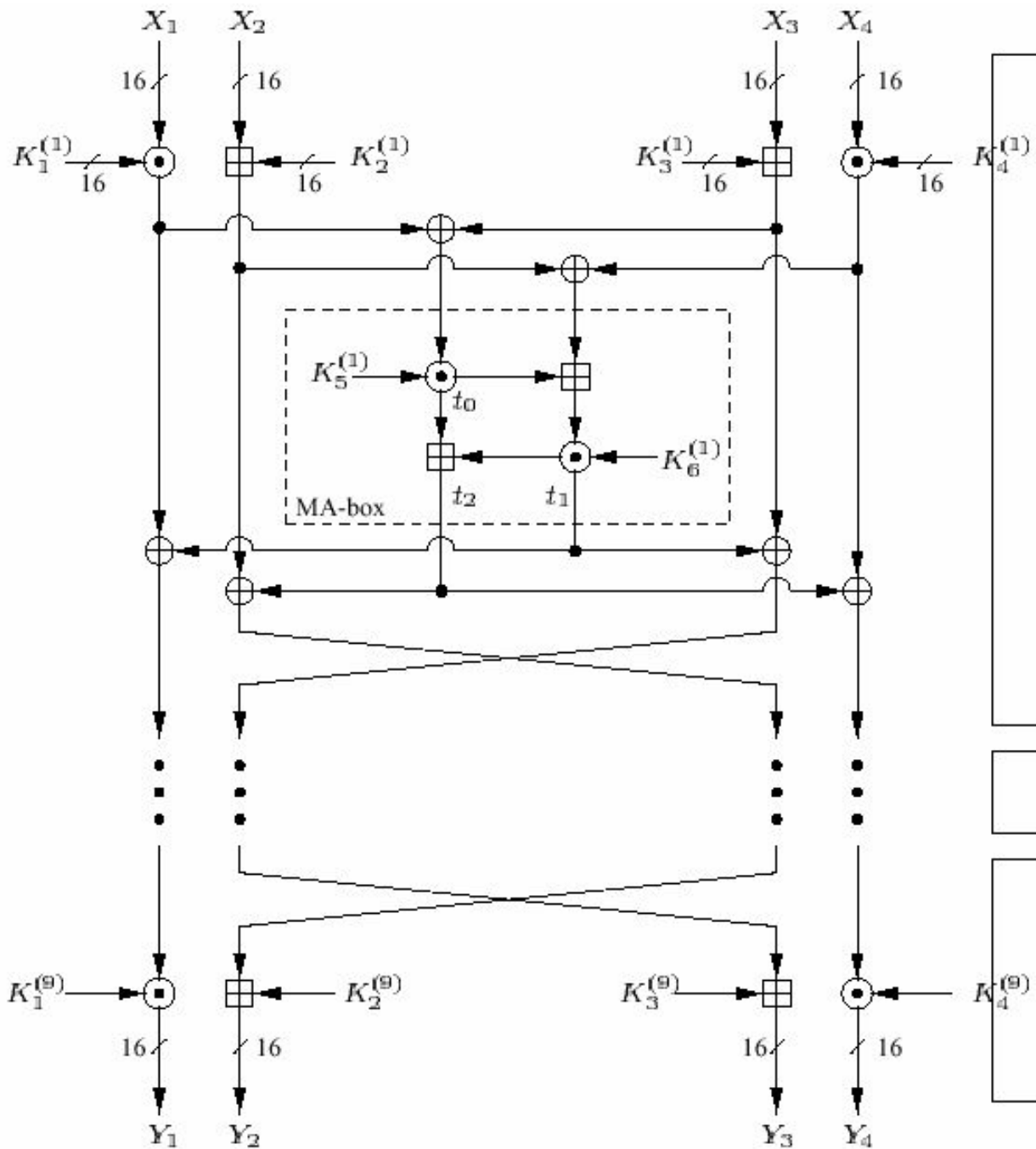
2 цикл -  $K1(2), K2(2), K3(2), K4(2), K5(2), K6(2)$ ,

3 цикл -  $K1(3), K2(3), K3(3), K4(3), K5(3), K6(3)$ ,

8 цикл -  $K1(8), K2(8), K3(8), K4(8), K5(8), K6(8)$ ,

выходное отображение -  $K1(9), K2(9), K3(9), K4(9)$ .

На каждом раунде используются только 96 бит подключа, множество бит ключа на каждой итерации не пересекаются, и не существует отношения простого сдвига между подключами разных раундов. Это происходит потому, что на каждом раунде используется только шесть подключей, в то время как при каждой ротации ключа получается восемь подключей.



**Схема алгоритма**  
**Открытый текст**  $x_1$  (16 бит),  $x_2$  (16 бит),  $x_3$  (16 бит),  $x_4$  (16 бит).

**Шаг 1.**  $c_1 = x_1 \{*\} k_1$   
**Шаг 2.**  $c_2 = x_2 \{+\} k_2$   
**Шаг 3.**  $c_3 = x_3 \{+\} k_3$   
**Шаг 4.**  $c_4 = x_4 \{*\} k_4$

**Шаг 5.**  $c_5 = c_1 \{+\} c_3$

**Шаг 6.**  $c_6 = c_2 \{+\} c_4$

**Шаг 7.**  $c_7 = c_5 \{*\} k_5$

**Шаг 8.**  $c_8 = c_6 \{+\} c_7$

**Шаг 9.**  $c_9 = c_8 \{*\} k_6$

**Шаг 10.**  $c_{10} = c_9 \{+\} c_7$

**Шаг 11.**  $c_{11} = c_1 \{+\} c_9$

**Шаг 12.**  $c_{12} = c_3 \{+\} c_9$

**Шаг 13.**  $c_{13} = c_2 \{+\} c_{10}$

**Шаг 14.**  $c_{14} = c_4 \{+\} c_{10}$

**Входные блоки**  
**следующего цикла –**  
 $c_{11}, c_{12}, c_{13}, c_{14}$ .

# Выходное отображение

После 8 цикла реализуется  
выходное отображение

Шаг 1.  $y_1 = x_1(9) \{*\} k_1(9)$

Шаг 2.  $y_2 = x_2(9) [+ ] k_2(9)$

Шаг 3.  $y_3 = x_3(9) [+ ] k_3(9)$

Шаг 4.  $y_4 = x_4(9) \{*\} k_4(9)$

Шифрованным текстом является  
блок  $y_1, y_2, y_3, y_4$ .

## Расшифровка

Метод вычисления, использующийся для расшифровки текста по существу такой же, как и при его шифровании. Единственное отличие состоит в том, что для расшифровки используются другие подключи. В процессе расшифровки подключи должны использоваться в обратном порядке. Первый и четвёртый подключи  $i$ -го раунда расшифровки получаются из первого и четвёртого подключа  $(10-i)$ -го раунда шифрования мультипликативной инверсией. Для 1-го и 9-го раундов второй и третий подключи расшифровки получаются из второго и третьего подключей 9-го и 1-го раундов шифрования аддитивной инверсией. Для раундов со 2-го по 8-й второй и третий подключи расшифровки получаются из третьего и второго подключей с 8-го по 2-й раундов шифрования аддитивной инверсией. Последние два подключа  $i$ -го раунда расшифровки равны последним двум подключам  $(9-i)$ -го раунда шифрования. Мультипликативная инверсия подключа  $K$  обозначается  $1/K$  и  $K^{-1}$ . Так как  $2^{16}+1$  простое число, каждое целое не равное нулю  $K$  имеет уникальную мультипликативную инверсию по модулю  $2^{16}+1$ . Аддитивная инверсия подключа  $K$  обозначается  $-K$ .

Аппаратная реализация

Аппаратная реализация имеет перед программной следующие преимущества:

**существенное повышение скорости шифрования** за счёт использования параллелизма при выполнении операций

**меньшее энергопотребление**

## Развитие аппаратных реализаций IDEA

1998 программная 23,53**Мб/сек** Limraa

2000 программная 44**Мб/сек** Limraa

1992 [ASIC](#) 1992 ASIC 1,5 мкм [КМОП](#) 44**Мб/сек** Bonnenberg и др.

1994 [ASIC](#) 1994 ASIC 1,2 мкм [КМОП](#) 177**Мб/сек** Curiger, Zimmermann и др.

1995 [ASIC](#) 1995 ASIC 0,8 мкм [КМОП](#) 355**Мб/сек** Wolter и др.

1998 [ASIC](#) 1998 ASIC 0,7 мкм [КМОП](#) 424**Мб/сек** Salomao и др.

1998 4 x XC4020XL 528**Мб/сек** Mencer и др.

1999 [ASIC](#) 1999 ASIC 0,25 мкм [КМОП](#) 720 **Мб/сек** Ascom

2000 Xilinx Virtex XCV300-6 1166**Мб/сек** Leong и др.

2000 [ASIC](#) 2000 ASIC 0,25 мкм [КМОП](#) 1013**Мб/сек** Goldstein и др.

В [2002 году](#) в 2002 году была опубликована работа о реализации IDEA на [ПЛИС](#) все той же фирмы Xilinx семейства Virtex-E. Устройство XCV1000E-6BG560 при частоте 105,9 МГц достигает скорости шифрования 6,78 Гб/сек.

Брюс Шнайер отозвался об IDEA так: «Мне кажется, это самый лучший и надежный блочный алгоритм, опубликованный до настоящего времени».

Существуют успешные атаки, применимые к IDEA с меньшим числом раундов (полный IDEA имеет 8.5 раундов). Успешной считается атака, если вскрытие шифра с её помощью требует меньшего количества операций, чем при полном переборе ключей. Метод вскрытия Вилли Майера (*Willi Meier*) оказался эффективнее вскрытия полным перебором ключей только для IDEA с 2 раундами. Лучшая атака на 2007 год применима ко всем ключам и может взломать IDEA с 6-ю раундами



## Сравнение с некоторыми блочными алгоритмами

Алгоритм	Размер ключа, бит	Длина блока, бит	Число раундов	Скорость шифрования на <a href="#">Intel486SX/33МГц</a> (Кбайт/с)
<a href="#">DES</a>	56	64	16	35
IDEA	128	64	8	70
<a href="#">Blowfish</a>	32-448	64	16	135
<a href="#">ГОСТ 28147-89</a>	256	64	32	53

# Преимущества и недостатки IDEA

## Преимущества

- В программной реализации на Intel486SX по сравнению с DES IDEA в два раза быстрее, что является существенным повышением скорости,
- длина ключа у IDEA имеет размер 128 бит, против 56 бит у DES, что является хорошим улучшением против полного перебора ключей.
- Вероятность использования слабых ключей очень мала и составляет  $2^{-64}$ .
- IDEA быстрее алгоритма ГОСТ 28147-89 (в программной реализации на Intel486SX).
- Использование IDEA в параллельных режимах шифрования на процессорах Pentium III и Pentium MMX позволяет получать высокие скорости. По сравнению с финалистами AES, 4-way IDEA лишь слегка медленнее, чем RC6 и Rijndael на Pentium II, но быстрее, чем Twofish и MARS. На Pentium III 4-way IDEA даже быстрее RC6 и Rijndael.
- Преимуществом также является хорошая изученность и устойчивость к общеизвестным средствам криптоанализа.

## Недостатки

- IDEA значительно медленнее, почти в два раза, чем Blowfish (в программной реализации на Intel486SX).
- Существенным недостатком является то, что IDEA запатентован, так как это препятствует его свободному распространению.
- IDEA не предусматривает увеличение длины ключа.
- Недостатком можно также считать тот факт, что не все работы по криптоанализу были опубликованы, то есть вполне возможно, что шифр взломан, или будет взломан в будущем.

## Применение IDEA

Алгоритм IDEA является торговой маркой и запатентован в Австрии, Франции, Германии, Италии, Нидерландах, Испании, Швеции, Швейцарии, Англии (Европейский патент EP-B-0482154), Соединенных штатах Америки (U.S. Patent 5 214 703, выпущен 25 Мая, 1993) и в Японии (JP 3225440).

Действие [патента](#) Алгоритм IDEA является торговой маркой и запатентован в Австрии, Франции, Германии, Италии, Нидерландах, Испании, Швеции, Швейцарии, Англии (Европейский патент EP-B-0482154), Соединенных штатах Америки (U.S. Patent 5 214 703, выпущен 25 Мая, 1993) и в Японии (JP 3225440). Действие патента истекает в 2010—2011 годах. Сегодня [лицензия](#) принадлежит компании MediaCrypt и позволяет свободно использовать алгоритм в некоммерческих приложениях.

Типичные области применения IDEA:

шифрование [аудио](#) шифрование аудио и [видео](#) шифрование аудио и видео данных для [кабельного телевидения](#) шифрование аудио и видео данных для кабельного телевидения, [видеоконференций](#) шифрование аудио и видео данных для кабельного телевидения, видеоконференций, [дистанционного обучения](#) шифрование аудио и видео данных для кабельного телевидения, видеоконференций, дистанционного обучения, [VoIP](#) защита коммерческой и [финансовой](#) защита коммерческой и финансовой информации, отражающей [конъюнктурные](#) колебания линии [связи](#) линии связи через [модем](#) линии связи через модем, [роутер](#) линии связи через модем, роутер или [ATM](#) линии связи через модем, роутер или ATM линию, [GSM](#) [технология](#) [смарт-карты](#)

# Недостатки DES

## Double DES

Наиболее логичным способом противодействия полному перебору ключа DES выглядит многократное зашифрование данных алгоритмом DES с различными ключами. Следующий алгоритм получил название Double DES – двойной DES:

$$C = E_{k2/2}(E_{k1/2}(M)),$$

где  $k1/2$  и  $k2/2$  – половины двойного ключа алгоритма Double DES, каждая из которых представляет собой обычный 56-битный ключ DES, а  $E$  – функция зашифрования блока данных обычным DES-ом.

Если бы при двойном шифровании DES выполнялось следующее свойство:

$$C = E_{k2/2}(E_{k1/2}(M)) = E_k(M),$$

для любых значений  $k1/2$  и  $k2/2$ , то двойное шифрование не приводило бы к усилению против полного перебора ключа – всегда нашелся бы такой ключ  $k$ , однократное зашифрование которым было бы эквивалентно двукратному шифрованию на ключах  $k1/2$  и  $k2/2$ , а для нахождения ключа  $k$  достаточно было бы перебрать 255 ключей. К счастью, DES не обладает таким свойством, что доказано в [20], поэтому Double DES действительно удваивает эффективный размер ключа – до 112 бит, а при современном развитии вычислительной техники полный перебор 112-битного ключа невозможен.

## **Атака «встреча посередине»**

предложена Ральфом Мерклем (Ralph Merkle) и Марином Хеллманом . С помощью этой атаки криптоаналитик может получить  $k_1/2$  и  $k_2/2$  при наличии всего двух пар открытого текста и шифртекста ( $M_1 - C_1$  и  $M_2 - C_2$ ) следующим образом:

**Шаг 1.** Выполняется зашифрование  $E_{k_x}(M_1)$  на всем ключевом пространстве с записью результатов в некоторую таблицу.

**Шаг 2.** Производится расшифрование  $D_{k_y}(C_1)$  также на всем ключевом пространстве; результаты расшифрования сравниваются со всеми записями в таблице, сформированной на шаге 1.

**Шаг 3.** Если какой-либо результат, полученный на шаге 2, совпал с одним из результатов шага 1, то можно предположить, что нужный ключ найден, т. е. соответствующие совпадающему результату  $k_x = k_1/2$ , а  $k_y = k_2/2$ . Однако таких совпадений может быть много, их количество оценивается в [1] как 248.

**Шаг 4.** Для отсеечения «ложных» ключей необходимо повторить предыдущие шаги с парой  $M_2 - C_2$ , сузив пространство перебора только до вариантов, приводящих к совпадениям (т. е. примерно 248). Вероятность наличия более чем одного совпадения после повторного перебора оценивается в [1] как 2–16. Такая атака, выполняющая, фактически, перебор половинок двойного ключа, как со стороны открытого текста, так и со стороны шифртекста, требует примерно в 2 раза больше вычислений, чем перебор обычного ключа DES, однако, требует также много памяти для хранения промежуточных результатов. Тем не менее, атака является реально осуществимой на практике, поэтому алгоритм Double DES не используется. Используется Triple DES

# Тройной DES

Достаточно медленно работает, так что некоторые приложения не могут работать по этим алгоритмом.

Так как текст, зашифрованный двойным DES оказывается не стойким при криптографической атаке то текст шифруется 3 раза DES. Таким образом длина ключа возрастает до 168-бит (56x3).

## Типы тройного шифрования DES:

DES-EEE3: Шифруется 3 раза с 3 различными ключами.

DES-EDE3: 3 DES операции шифровка-расшифровка-шифровка с 3 различными ключами.

DES-EEE2 и DES-EDE2: Как и предыдущие, за исключением того, что первая и третья операции используют одинаковый ключ.

*Тройной DES* является достаточно популярной альтернативой *DES* и используется при управлении ключами в стандартах ANSI X9.17 и ISO 8732 и в PEM (Privacy Enhanced Mail).

Известных криптографических атак на *тройной DES* не существует. Цена подбора ключа в *тройном DES* равна  $2^{112}$ .

# DESX

Метод DESX создан Рональдом Ривестом и формально продемонстрирована Killian и Rogaway. Этот метод — усиленный вариант DES. DESX отличается от DES тем, что каждый бит входного открытого текста DESX логически суммируется по модулю 2 с 64 битами дополнительного ключа, а затем шифруется по алгоритму DES. Каждый бит результата также логически суммируется по модулю 2 с другими 64 битами ключа.

Таким образом, длина ключа увеличивается до  $56 + 2 \times 64 = 184$  бит.

Главной причиной использования DESX является простой в вычислительном смысле способ значительного повысить стойкость DES к атакам полного перебора ключа.

Скорость алгоритма DESX приблизительно равна скорости DES. DESX достаточно стоек, имеет аппаратную реализацию и широко используется.

Реализация DESX включена в криптографические библиотеки BSAFE компании RSA Security с конца 80х годов.



**AES**

# Недостатки DES

В 80-х годах в США был принят стандарт симметричного криптоалгоритма для внутреннего применения DES (Data Encryption Standard).

В настоящее время DES устарел по двум причинам :

- 1) Малая длина его ключа (56 бит).
- 2) Алгоритм ориентирован на аппаратную реализацию, то есть содержит операции, выполняемые на микропроцессорах за неприемлемо большое время (например, такие как перестановка бит внутри машинного слова по определенной схеме).

## Требования, предъявленные к кандидатам на AES в 1998 году

NIST – National Institute of Standards & Technology объявил конкурс на новый стандарт симметричного криптоалгоритма

стандарт блочных шифров США с 2000 года

- 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 рекомендации

- использовать операции, легко реализуемые как аппаратно, так и программно,
- ориентироваться на 32-разрядные процессоры,
- не усложнять без необходимости структуру шифра

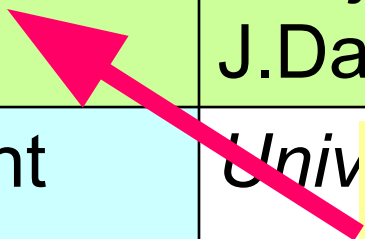
На первом этапе в оргкомитет соревнования поступило 15 заявок из совершенно разных уголков мира. В течение 2 лет специалисты комитета, исследуя самостоятельно, и изучая публикации других исследователей, выбрали 5 лучших представителей, прошедших в "финал" соревнования.

Полное описание всех 15 алгоритмов претендентов на AES, включая исследования по их криптостойкости представлены на сервере института NIST.

<b>Алгоритм</b>	<b>Автор</b>	<b>Страна</b>	<b>Быстродействие (asm, 200МГц)</b>
MARS	IBM	US	8 Мбайт/с
RC6	R.Rivest & Co	US	12 Мбайт/с
Rijndael	V.Rijmen & J.Daemen	BE	7 Мбайт/с
Serpent	<i>Universities</i>	IS, UK, NO	2 Мбайт/с
TwoFish	B.Schneier & Co	US	11 Мбайт/с

Все эти алгоритмы были признаны достаточно стойкими и успешно противостоящими всем широко известным методам криптоанализа.

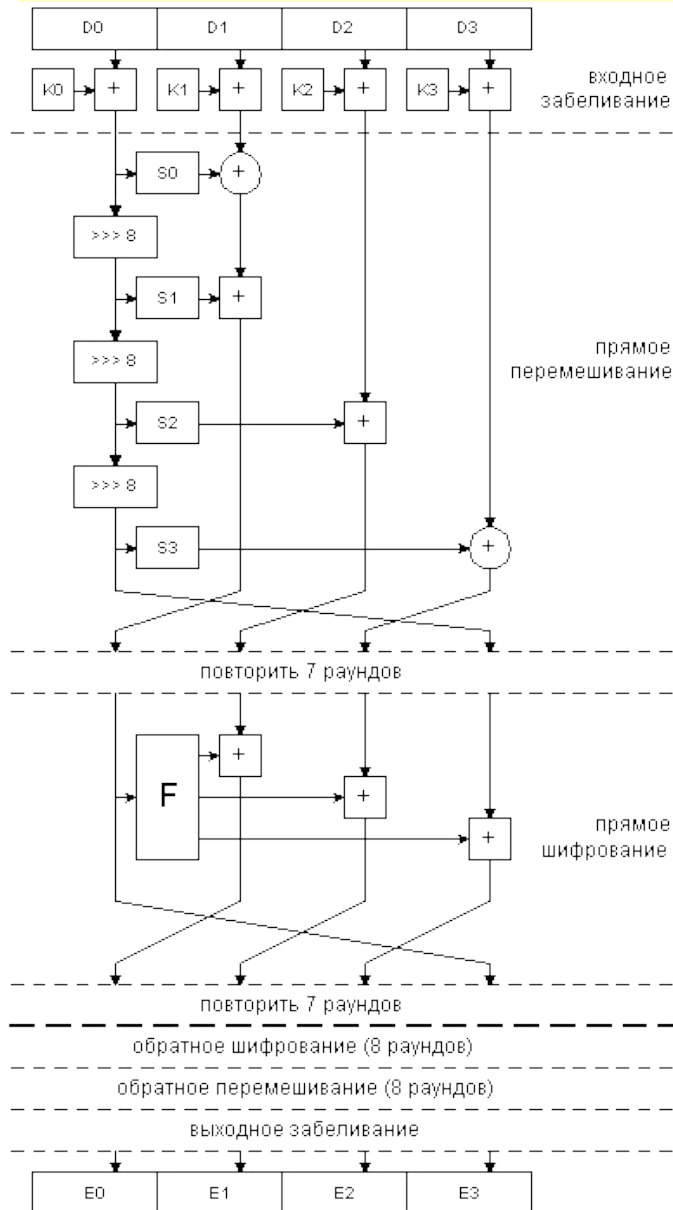
Алгоритм	Автор	Страна	Быстродействие (asm, 200МГц)
MARS	IBM	US	8 Мбайт/с
RC6	R.Rivest & Co	US	12 Мбайт/с
Rijndael	V.Rijmen & J.Daemen	BE	7 Мбайт/с
Serpent	Univ		
TwoFish	B.Sc & Co		



2 октября 2000 года NIST объявил о своем выборе – победителем конкурса стал бельгийский алгоритм RIJNDAEL. С этого момента с алгоритма-победителя сняты все патентные ограничения – его можно будет использовать в любой криптопрограмме без отчисления каких-либо средств создателю.

С

# Финалист AES – шифр MARS



1 - входное забеливание : ко всем байтам исходного текста добавляются байты из материала ключа.

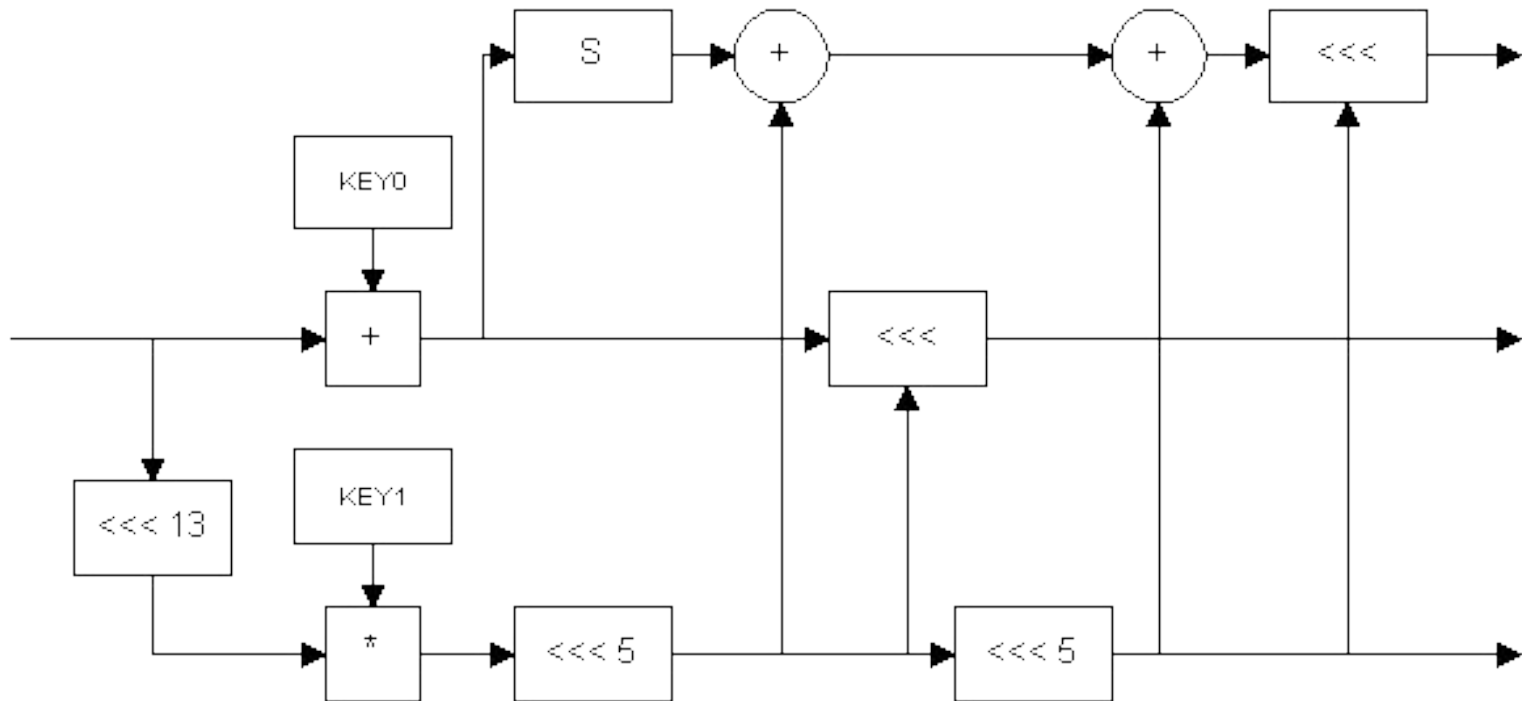
2 - прямое перемешивание, сеть Фейстеля (8 раундов) без добавления ключа.

3 - сеть Фейстеля третьего типа с 4 ветвями (8 раундов).

Затем повторяются те же операции, но в обратном порядке : сначала шифрование, перемешивание, забеливание. При этом во вторые варианты всех операций внесены некоторые изменения таким образом, чтобы криптоалгоритм в целом стал абсолютно симметричным. То есть, в алгоритме MARS для любого X выполняется выражение  $EnCrypt(EnCrypt(X))=X$



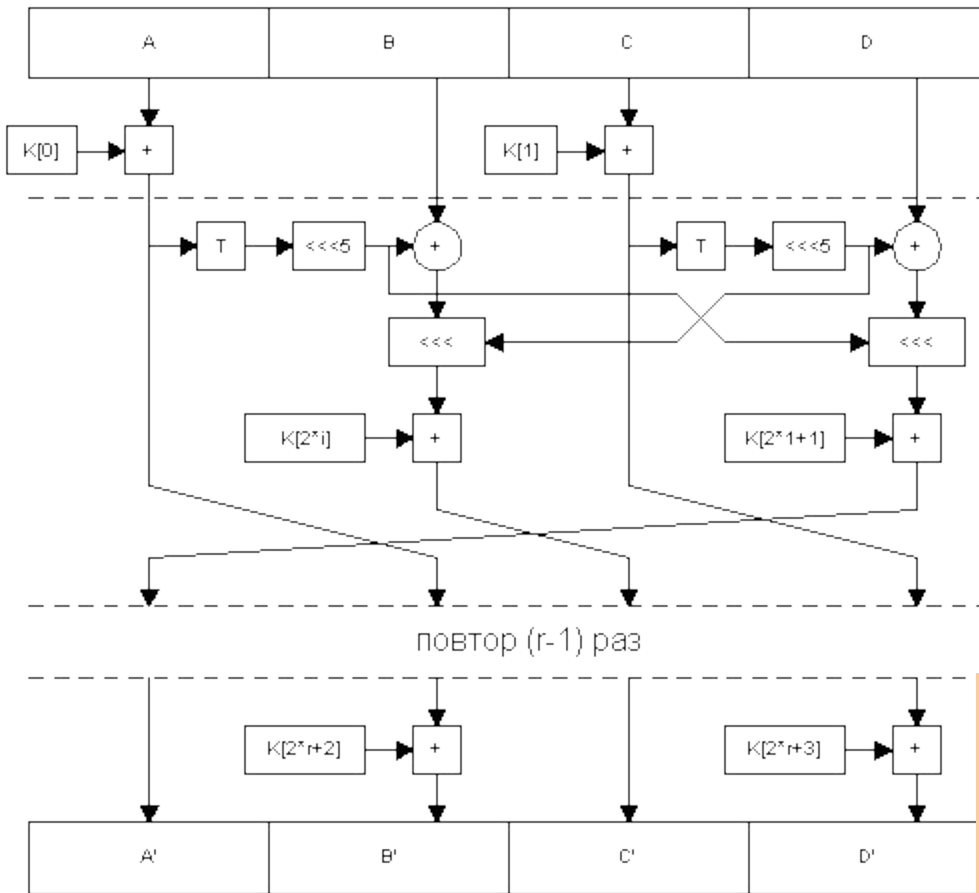
## Функция F:



В алгоритме MARS использованы практически все виды операций, применяемых в криптографических преобразованиях : сложение, "исключающее ИЛИ", сдвиг на фиксированное число бит, сдвиг на переменное число бит, умножение и табличные подстановки.

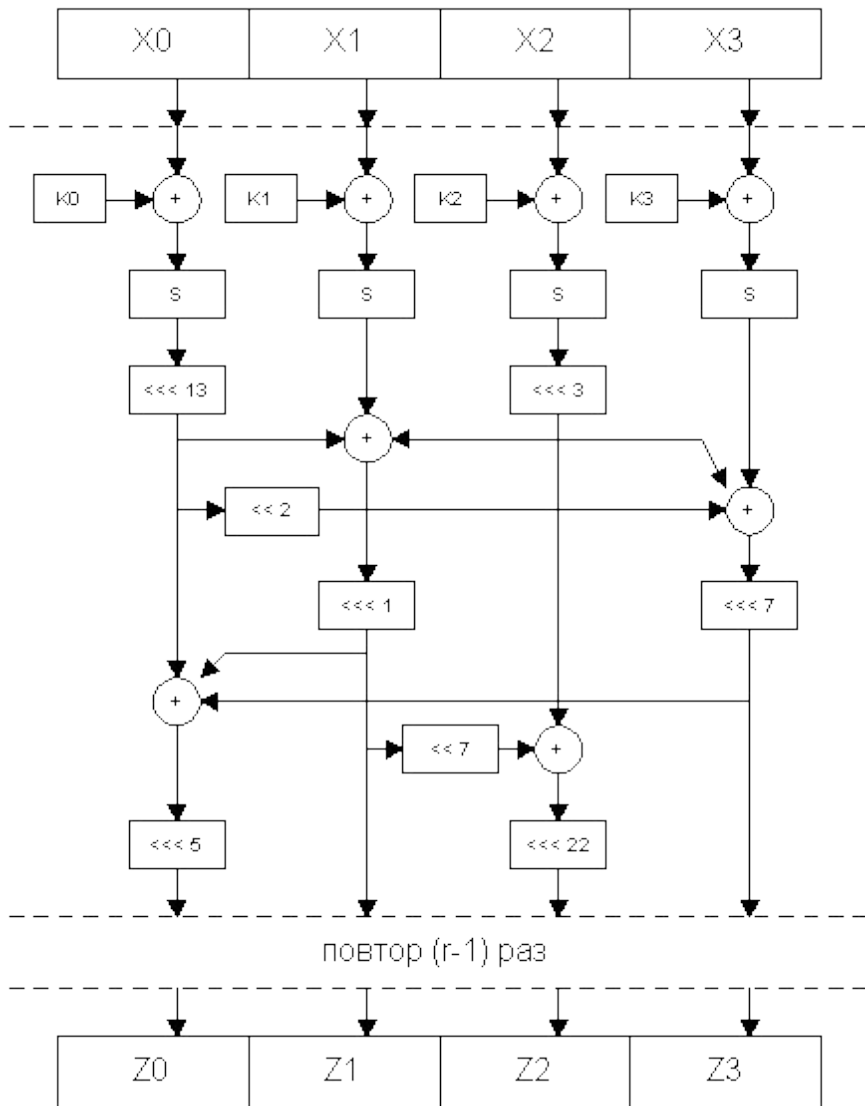
# Финалист AES – шифр RC6

- продолжение RC5, разработанного Рональдом Ривестом. RC5 был незначительно изменен для того, чтобы соответствовать требованиям AES по длине ключа и размеру блока. При этом алгоритм стал еще быстрее, а его ядро, унаследованное от RC5, имеет солидный запас исследований, проведенных задолго до объявления конкурса AES.



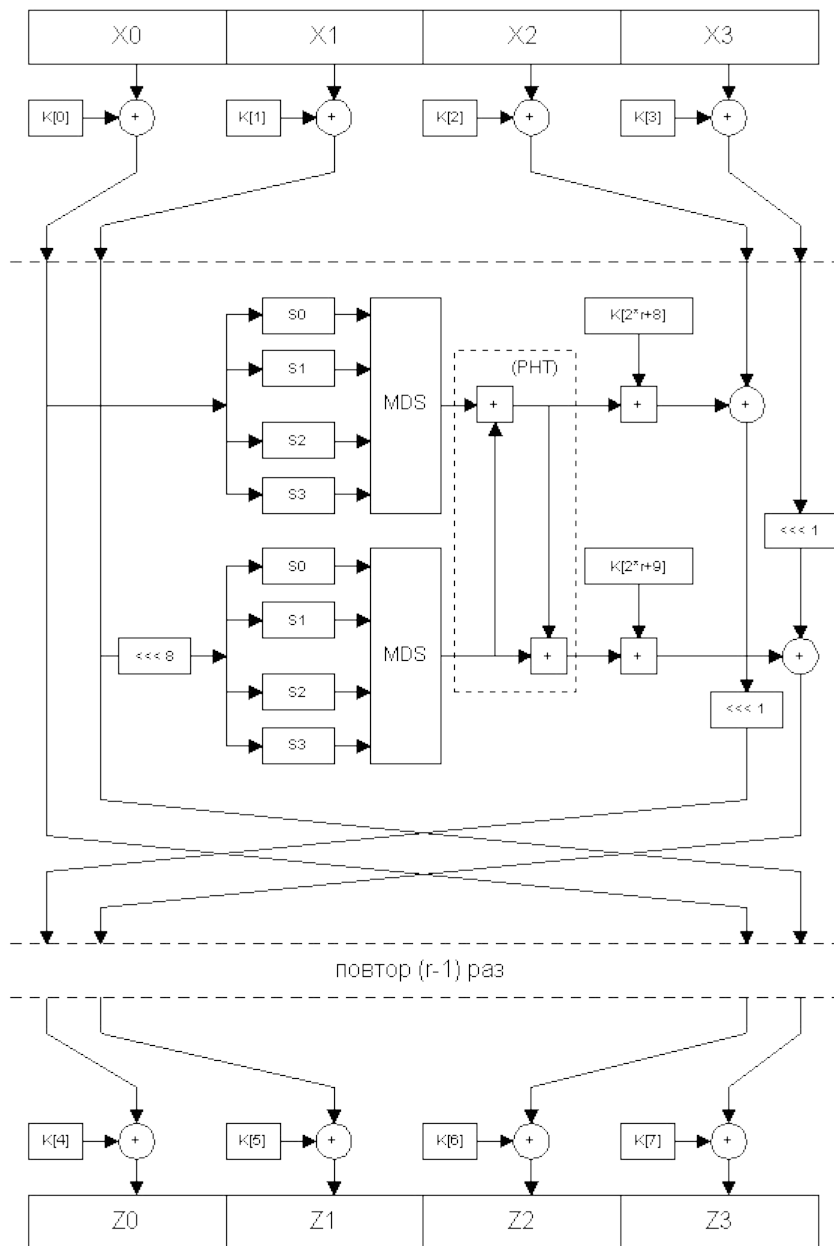
сеть Фейштеля с 4 ветвями. Разработчики рекомендуют при шифровании использовать 20 раундов сети, хотя в принципе их количество не регламентируется. При 20 повторах операции шифрования алгоритм имеет самую высокую скорость среди 5 финалистов AES.

# Финалист AES – шифр Serpent



Алгоритм разработан группой ученых из нескольких исследовательских центров мира. Алгоритм представляет собой сеть Фейстеля для четырех ветвей. Используются только исключающее "ИЛИ", табличные подстановки и битовые сдвиги. Алгоритм состоит из 32 раундов.

# Финалист AES – шифр TwoFish



Алгоритм разработан компанией Counterpain Security Systems, возглавляемой Брюсом Шнайером. Предыдущая программная разработка этой фирмы, называвшаяся BlowFish, до сих пор является признанным криптостойким алгоритмом Сеть Фейштеля. Единственным нарицанием, поступившим в адрес TwoFish от независимых исследователей, является тот факт, что при расширении материала ключа в алгоритме используется сам же алгоритм. Двойное применение блочного шифра довольно сильно усложняет его анализ на предмет **наличия слабых ключей или недокументированных замаскированных связей между входными и выходными данными.**

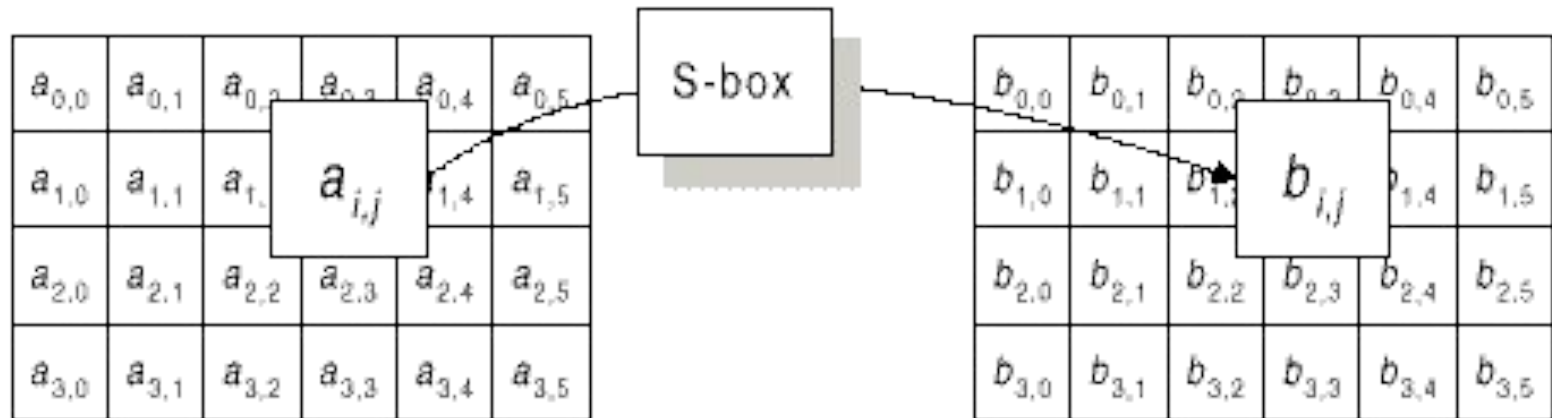
## Победитель AES – шифр Rijndael

Не использует сеть Фейстеля для криптопреобразований. Алгоритм представляет каждый блок кодируемых данных в виде двумерного массива байт размером 4x4, 4x6 или 4x8 в зависимости от установленной длины блока. Далее на соответствующих этапах преобразования производятся либо над независимыми столбцами, либо над независимыми строками, либо вообще над отдельными байтами в таблице.

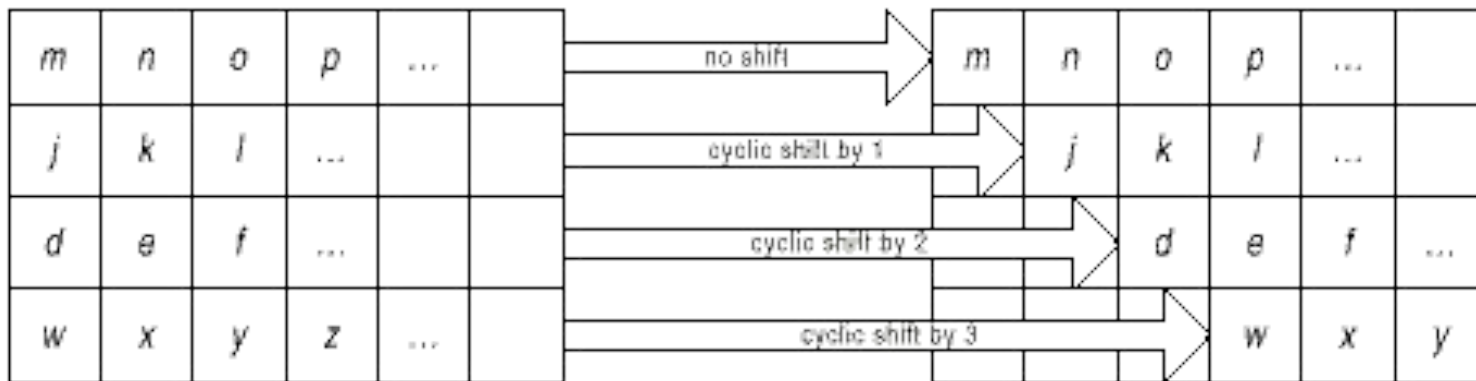
Все преобразования в шифре имеют строгое математическое обоснование. Сама структура и последовательность операций позволяют выполнять данный алгоритм эффективно как на 8-битных так и на 32-битных процессорах. В структуре алгоритма заложена возможность параллельного исполнения некоторых операций, что на многопроцессорных рабочих станциях может еще поднять скорость шифрования в 4 раза.

Алгоритм состоит из некоторого количества раундов (от 10 до 14 – это зависит от размера блока и длины ключа), в которых последовательно выполняются следующие операции :

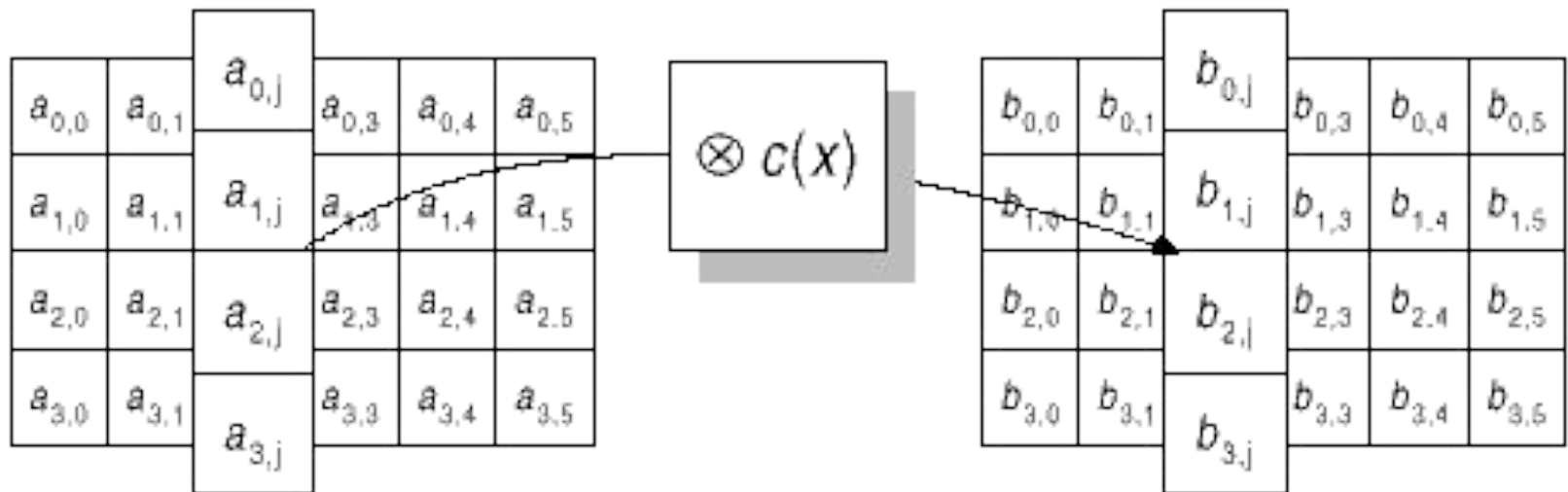
- 1. ByteSub – табличная подстановка 8x8 бит



## 2. ShiftRow – сдвиг строк в двумерном массиве на различные смещения

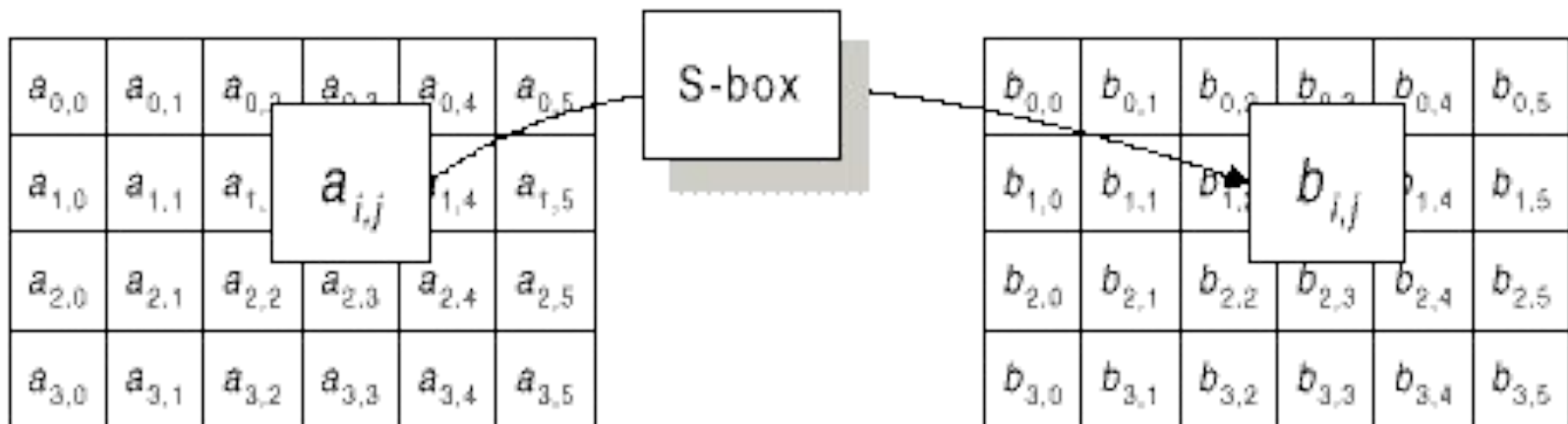


### 3. MixColumn – перемешивание данных внутри столбца





# AddRoundKey – добавление материала ключа операцией XOR



В последнем раунде операция перемешивания столбцов отсутствует, что делает всю последовательность операций симметричной.

# Криптоанализ AES

На презентации алгоритма разработчики продемонстрировали атаку на 6 раундов алгоритма и заявили о необходимости 10-14 раундов шифрования (в зависимости от размера ключа). В процессе отбора кандидатов механизм атак был усовершенствован до 7 раундов для 128-битных ключей, 8 раундов для 196-битных ключей и 9 раундов для 256-битных ключей. В результате остается от 3 до 5 раундов на обеспечение безопасности. Однако даже возможное развитие данных атак на 100% шифра потребовало бы  $2^{120}$  шагов и  $2^{100}$  байт памяти. Подобную атаку в настоящее время невозможно осуществить на практике и ориентировочно в ближайшие 50 лет.

# Криптоанализ Serpent

Наиболее консервативный из всех участников конкурса. Полностью ориентирован на обеспечение безопасности. Наилучшая из атак способна взломать 10 из 32 раундов. Главный недостаток – скорость ( в 3 раза медленнее AES, примерно равная DES). Иначе он с большой вероятностью занял бы 1 место благодаря своей консервативности.

# Криптоанализ Twofish

Почти такой же быстрый, как AES, но с большим запасом прочности. Наилучшая атака – 8 раундов из 16. Недостаток – длительность смены ключа шифрования.

# Криптоанализ RC6

Успешная атака -17 из 20 раундов.

# Криптоанализ MARS

Ошибка в коде программы привела к тому, что сгенерированная S-матрица не удовлетворяла условиям, выдвинутым самими разработчиками. В аргументах, приводимых авторами как доказательство устойчивости к линейному криптоанализу, обнаружались серьезные упущения.