Жуков А. Е.

Основные достижения теоретической криптологии в 2002 году (слайды)

AES ⇔ Rijndael

В 1997 г. Национальный Институт Стандартов и Технологий США (NIST)[1)] объявил о начале конкурса на новый стандарт криптографической защиты данных – ***AES*** (Advanced Encryption Standard). Хотя существующий с 1977 г. криптоалгоритм DES и остается действующим стандартом, он считается устаревшим по многим параметрам, среди которых малая длина ключа[2)], неудобство реализации на современных процессорах, малое быстродействие.

Несомненным достоинством алгоритма DES является его стойкость. За 25 лет интенсивного криптоанализа не было найдено методов вскрытия этого шифра, существенно отличающихся по эффективности от полного перебора всех ключей ключевого пространства. К новому стандарту были предъявлены следующие требования:

- криптоалгоритм должен быть симметричным блочным шифром, допускающим размеры ключей в 128, 192 или 256 бит;
- криптоалгоритм должен быть удобен как для аппаратной, так и для программной реализации;
- криптоалгоритм должен удовлетворять современным требованиям по следующим параметрам: стойкость, скорость, стоимость, гибкость.

[1)] Прежнее название – Национальное Бюро Стандартов.

[2)] DES остается стандартом лишь в качестве составной части алгоритма TDEA (Triple DES) со 112- или 168-битным ключом.

В конце 2000 г. победителем конкурса был объявлен криптоалгоритм Rijndael. С мая 2002 г. этот алгоритм с небольшими модификациями о которых будет сказано ниже, стал официальным стандартом шифрования данных AES. Материалы конкурса AES, в том числе документацию по алгоритмам – финалистам конкурса, можно найти на Web-сайте [http://www.nist.gov/encryption/aes/].

Общие сведения

AES – это итерационный блочный шифр, имеющий следующие параметры:

- длина информационного блока – 128 бит[3)];
- длина ключа – варьируется и может равняться 128, 192 или 256 битам;
- число циклов шифрования зависит от длины ключа и равняется 10, 12 или 14 циклам.

Промежуточные результаты преобразований, выполняемых над информационным блоком в рамках криптоалгоритма, называются в дальнейшем ***состояниями*** (*state*). Состояние рассматривается в виде прямоугольного (4×4)-массива байтов, по 4 байта в столбце. Цикловые ключи также представляются в виде (4×4)-массива байтов. На рис. 0.1. изображены состояние и цикловой ключ и указывается нумерация байтов в соответствующих массивах.

[3)] Основное отличие алгоритма Rijndael от принятого стандарта AES состоит в том, что алгоритм Rijndael допускает также информационные блоки длиной в 192 или 256 бит. Алгоритм выработки цикловых ключей и число циклов шифрования в Rijndael изменяется в зависимости от выбранной длины информационного блока.

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}

k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

Входные данные для шифра обозначаются как байты состояния в порядке a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, a_{21}, a_{31}, a_{41} ... После завершения действия шифра выходные данные получаются из байтов состояния в том же порядке.

Число циклов N_r зависит от значений N_b и N_k :

Зависимость числа циклов от числа столбцов в таблице состояний и ключа шифрования			
N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	**10**	**12**	**14**
$N_k = 6$	**12**	**12**	**14**
$N_k = 8$	**14**	**14**	**14**

Round **(*State*, *RoundKey*):**

***ByteSub*(*State*); // замена байтов**
***ShiftRow*(*State*); // сдвиг строк**
***MixColumn*(*State*); // перемешивание столбцов**
***AddRoundKey*(*State*, *RoundKey*);// добавление циклового ключа**

ByteSub *(Замена байтов)*

Преобразование ByteSub является S-блоком и представляет собой нелинейную замену байтов, выполняемую независимо для каждого байта состояния, т.е. преобразование *ByteSub* действует на каждый байт состояния**.**

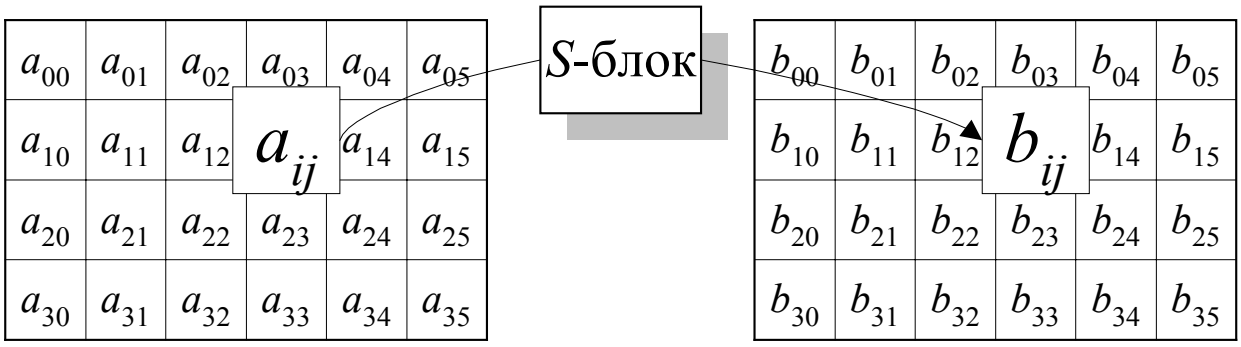


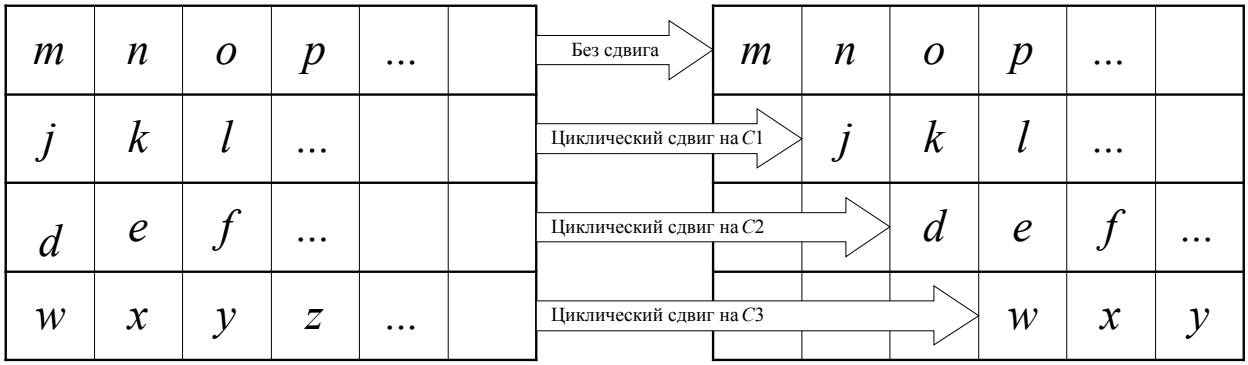
Таблица замены S-блока является инвертируемой и построена как композиция двух преобразований:

1. **переход к обратному элементу относительно умножения в поле GF(2^8), при этом нулевой элемент '00h' переходит сам в себя;**
2. **применение аффинного преобразования над 8-мерным двоичным вектором:**

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

ShiftRow *(Сдвиг строк)*

Строки матрицы состояния (кроме нулевой строки) циклически сдвигаются на различное число байт. Строка 1 сдвигается на *C*1 байт, строка 2 - на *C*2 байт и строка 3 - на *C*3 байт. Операция сдвига последних 3 строк состояния на определенную величину обозначена как ShiftRow (State).

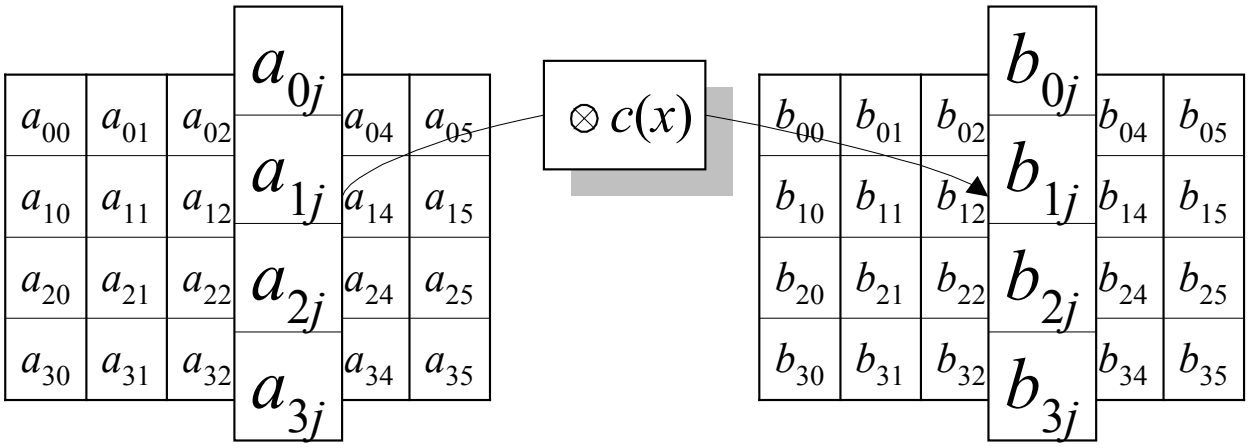


Значения сдвигов *C*1, *C*2 и *C*3 зависят от длины блока N_b .

Величина сдвига для разной длины блоков			
N_b	***C*1**	***C*2**	***C*3**
4	**1**	**2**	**3**
6	**1**	**2**	**3**
8	**1**	**3**	**4**

MixColumn *(Перемешивание столбцов)*

В этом преобразовании столбцы состояния рассматриваются как многочлены над полем GF(2^8) и умножаются по модулю $M(x) = x^4 + 1$ на многочлен $c(x) = '03h'x^3 + '01h'x^2 + '01h'x + '02'.$ Коэффициенты этого многочлена являются элементами поля GF(2^8) и приведены в шестнадцатеричной записи: '03h'=(00000011). Многочлен $c(x)$ взаимно прост с многочленом $M(x)$ и, следовательно, операция умножения на $c(x)$ по модулю $M(x)$ обратима. Многочлен, обратный к $c(x)$ по модулю $x^4 + 1$, равен $d(x) = '0Bh'x^3 + '0Dh'x^2 + '09h'x + '0Eh'$.



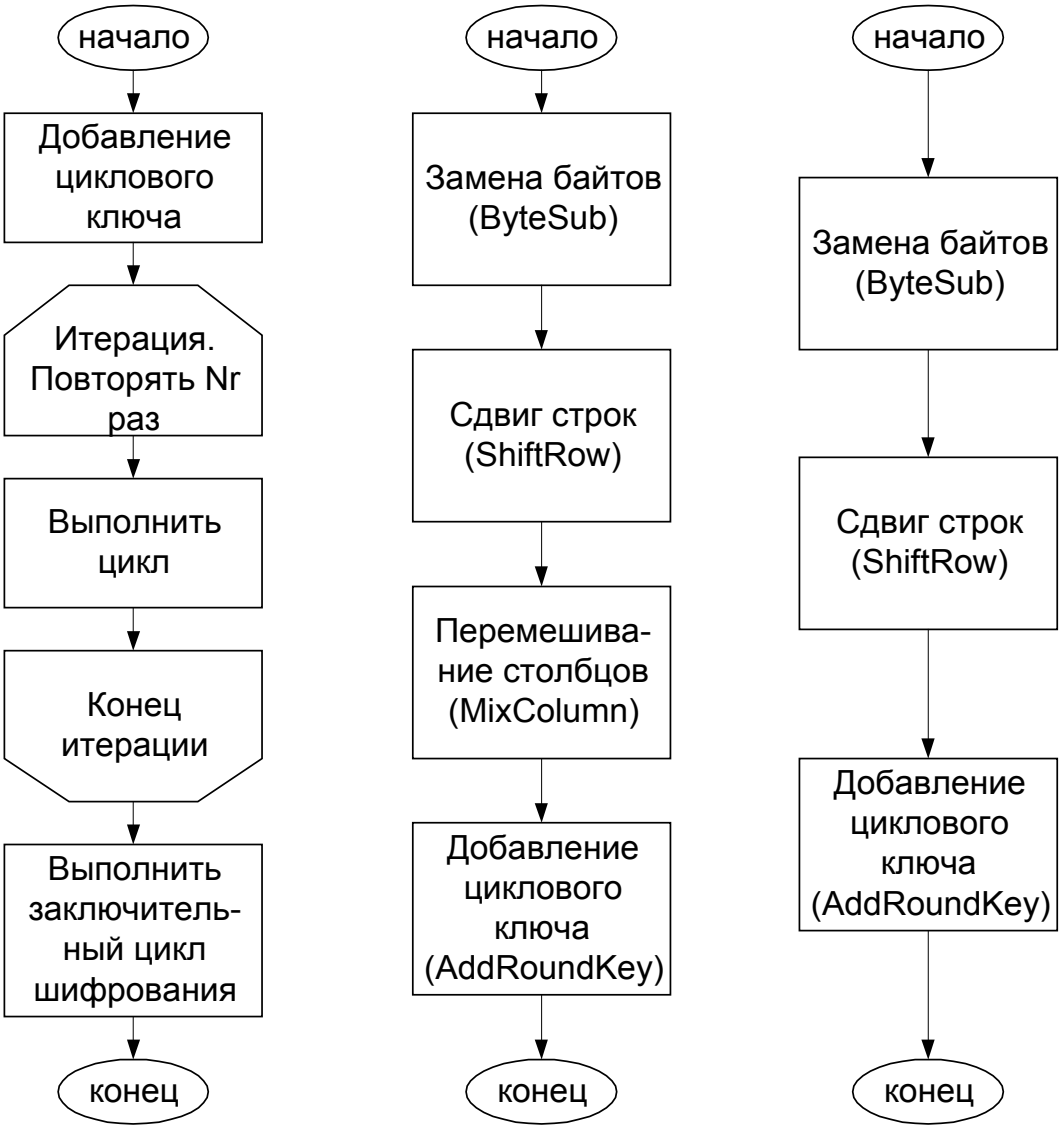
Это преобразование может быть также представлено в матричном виде следующим образом

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{pmatrix} 02h & 03h & 01h & 01h \\ 01h & 02h & 03h & 01h \\ 01h & 01h & 02h & 03h \\ 03h & 01h & 01h & 02h \end{pmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

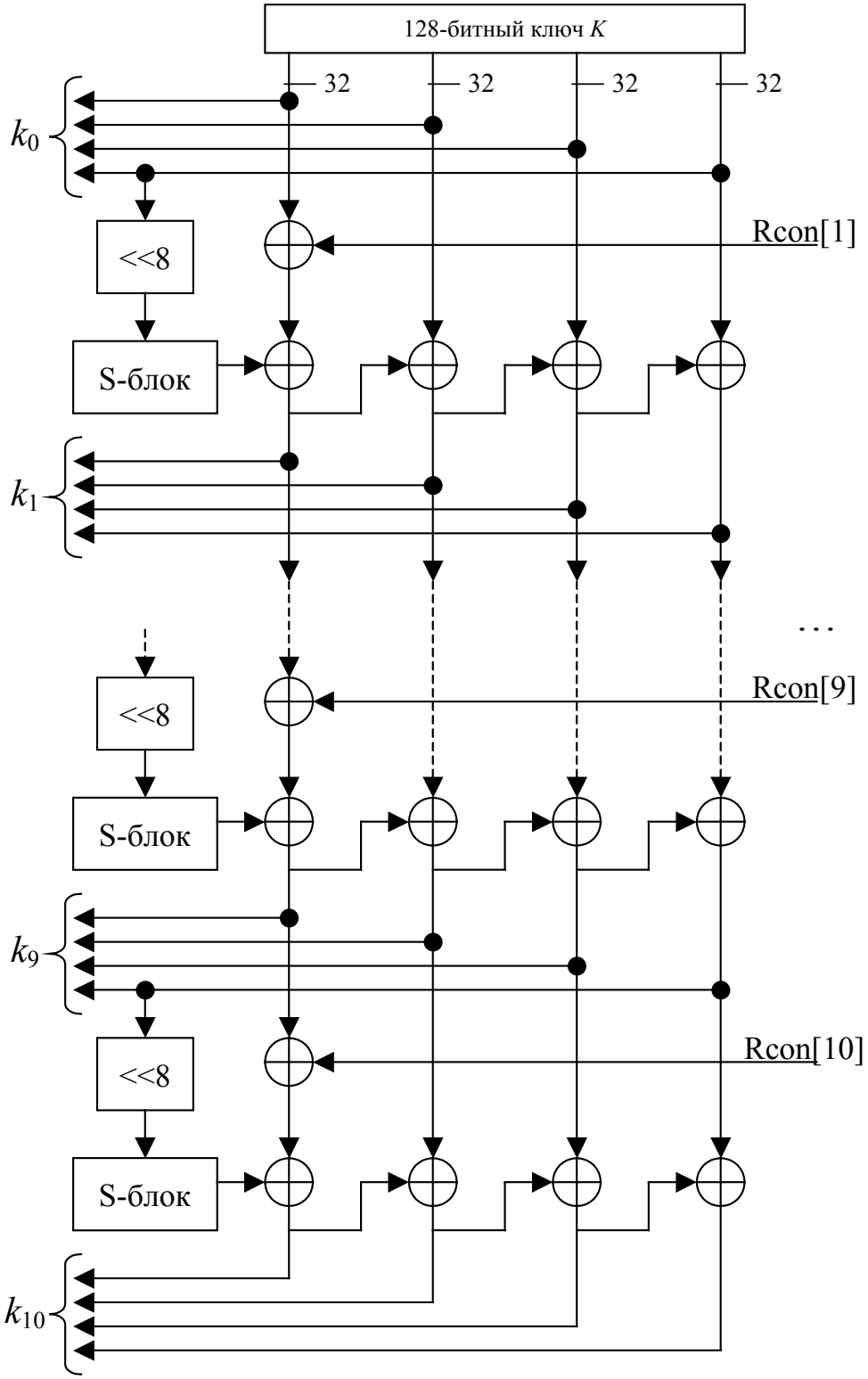
AddRoundKey *(Добавление циклового ключа)*

a_{00}	a_{01}	a_{02}	a_{03}	$\oplus$	k_{00}	k_{01}	k_{02}	k_{03}	$=$	b_{00}	b_{01}	b_{02}	b_{03}
a_{10}	a_{11}	a_{12}	a_{13}		k_{10}	k_{11}	k_{12}	k_{13}		b_{10}	b_{11}	b_{12}	b_{13}
a_{20}	a_{21}	a_{22}	a_{23}		k_{20}	k_{21}	k_{22}	k_{23}		b_{20}	b_{21}	b_{22}	b_{23}
a_{30}	a_{31}	a_{32}	a_{33}		k_{30}	k_{31}	k_{32}	k_{33}		b_{30}	b_{31}	b_{32}	b_{33}

Структура алгоритма



Расширение ключа (*Key Expansion*)



Первые результаты анализа

Шифр	Ключ	Объем материала	Временная сложность
AES-6	128-256	2^{32}	2^{72}
AES-6	128-256	$6*2^{32}$	2^{44}
AES-7	192	$19*2^{32}$	2^{155}
AES-7	256	$21*2^{32}$	2^{172}
AES-7	128-256	2^{128}	2^{120}
AES-8	192	2^{128}	2^{188}
AES-8	256	2^{128}	2^{204}
AES-9	256	2^{85}	2^{224}