

ПЕРЕВОД ТЕХНИЧЕСКОЙ ЧАСТИ СТАНДАРТА FIPS 197 (AES)

**ADVANCED ENCRYPTION STANDARD**  
(РАСШИРЕННЫЙ СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ)

**ВЕРСИЯ ДОКУМЕНТА 1.0**

ПЕРЕВОДЧИК – СТУДЕНТ МГТУ ИМ. БАУМАНА БАУЛИН ДМИТРИЙ

ПЕРЕВОДИЛ – СТАРАЛСЯ, НО ЗА КОСЯКИ НЕ ОТВЕЧАЮ )))

МОСКВА, 20.09.2007

## Оглавление

<b>1. Введение</b> .....	3
<b>2. Определения</b> .....	3
2.1 Словарь определений и сокращений.....	3
2.2 Параметры, символы и функции алгоритма.....	4
<b>3. Условные знаки и соглашения</b> .....	5
3.1 Входы и выходы.....	5
3.2 Байты.....	5
3.3 Массивы байт.....	6
3.4 Матрица состояния.....	6
3.5 Матрица состояния как массив столбцов.....	7
<b>4. Математическое введение</b> .....	7
4.1 Сложение.....	7
4.2 Умножение.....	8
4.2.1 Умножение на $x$ .....	9
4.3 Многочлены с коэффициентами, принадлежащими полю $GF(2^8)$ .....	9
<b>5. Описание алгоритма</b> .....	11
5.1 Шифрование (процедура Chipher).....	11
5.1.1 Процедура SubBytes.....	12
5.1.2 Процедура ShiftRows.....	14
5.1.3 Процедура MixColumns.....	14
5.1.4 Процедура AddRoundKey.....	15
5.2 Получение подключей (процедура KeyExpansion).....	16
5.3 Расшифрование (процедура InvChipher).....	17
5.3.1 Процедура InvShiftRows.....	18
5.3.2 Процедура InvSubBytes.....	19
5.3.3 Процедура InMixColumns.....	19
5.3.4 Инверсия процедуры AddRoundKey.....	20
5.3.5 Эквивалентная процедура расшифрования.....	20
<b>6. Вопросы реализации</b> .....	23
6.1 Требования к длине ключа.....	23
6.2 Ограничения на выбор ключа.....	23
6.3 Выбор длины ключа, размера блока и числа раундов.....	23
6.4 Рекомендации по реализации алгоритма на различных платформах.....	23
<b>Приложение А – пример получения подключей</b> .....	24
А.1 Расширение 128-ми битного ключа.....	24
А.2 Расширение 192-х битного ключа.....	25
А.3 Расширение 256-ти битного ключа.....	27
<b>Приложение Б – пример шифрования</b> .....	29
<b>Приложение В – примеры промежуточных вычислений</b> .....	31
В.1 – AES-128.....	32
В.2 – AES-192.....	35
В.3 – AES-256.....	38
<b>Приложение Г – ссылки</b> .....	43

## 1. Введение.

Этот стандарт описывает алгоритм Rijndael – симметричный блочный шифр, который обрабатывает данные блоками по 128 бит, используя ключ длиной 128, 192 или 256 бит. Алгоритм Rijndael был спроектирован для работы с более длинными блоками данных и ключами, но эта возможность не была использована в данном стандарте.

Далее описываемый алгоритм будем называть «алгоритм AES». Алгоритм может быть использован с ключами различной длины (128, 192 и 256 бит). Эти три разновидности алгоритма будем далее называть «AES-128», «AES-192» и «AES-256».

Кроме введения описание содержит следующие разделы:

- 2 – определение терминов, сокращений, параметров алгоритма, символов и функций
- 3 – условные знаки и соглашения, используемые при описании алгоритма, включая последовательность и нумерацию бит, байт и слов
- 4 – математические сведения, необходимые для понимания алгоритма
- 5 – описание алгоритма
- 6 – вопросы реализации, такие как поддерживаемая длина ключей, ограничения при выборе ключей, а также возможные значения размера блока/ключа и числа раундов.

К стандарту прилагается несколько приложений, в которых шаг за шагом показаны примеры работы процедуры расширения ключа, шифрования, расшифрования, и ссылки.

## 2. Определения

### 2.1 Словарь определений и сокращений

В данном стандарте используются следующие определения:

**AES** – Расширенный стандарт шифрования данных

**Аффинное преобразование** – преобразование, заключающееся в умножении вектора на матрицу с последующим сложением результата умножения с другим вектором.

**Массив** – пронумерованная последовательность одинаковых элементов (например массив байтов).

**Бит** – двоичная цифра, принимающая значение 0 или 1.

**Блок** – последовательность бит, представляющих входные данные, выходные данные, матрицу состояния и ключи раундов. Количество битов в последовательности называется длиной последовательности. Блок также можно рассматривать как массив байт.

**Байт** – группа из 8-ми бит, которая обрабатывается либо как единое целое, либо как массив из 8-ми отдельных бит

**Шифрование** – серия преобразований, которая преобразует открытый текст в шифртекст, используя ключ шифрования

**Ключ шифрования** – секретный криптографический ключ, используемый в процедуре расширения ключа для получения промежуточных ключей (ключей раундов). Ключ может быть представлен как прямоугольная матрица байт, имеющая 4 строки и столбцов.

**Шифртекст** – выходные данные операции шифрования и входные данные для операции расшифрования.

**Расшифрование** – серия операций, преобразующих шифртекст в открытый текст, используя ключ шифрования.

**Расширение ключа** – процесс для генерации промежуточных ключей (ключей раундов) из ключа шифрования.

**Открытый текст** – данные, являющиеся входными для процедуры шифрования и выходными для процедуры расшифрования.

**Rijndael** – криптографический алгоритм, описываемый в данном стандарте

**Ключи раундов (подключи)** – это числа, получаемые из ключа шифрования с помощью процедуры расширения ключа. Ключи раундов используются вместе с матрицей состояния при шифровании и расшифровании данных

**Матрица состояния (состояние)** – промежуточный результат шифрования, который может быть представлен как прямоугольная матрица байт, имеющая 4 строки и  $N_b$  столбцов.

**S-блок** – нелинейная таблица подстановки, используемая в преобразованиях подстановки и в процедуре расширения ключа для замены байта на байт

**Слово** – группа из 32 бит, которая обрабатывается либо как единое целое, либо как массив из 4-х байт.

## 2.2 Параметры, символы и функции алгоритма

В данном стандарте используются следующие параметры, символы и функции:

**AddRoundKey()** – преобразование при шифровании и расшифровании, заключающееся в сложении ключа раунда с матрицей состояния с помощью операции XOR. Длина ключа раунда равна длине состояния (то есть для  $N_b = 4$  длина ключа раунда равна 128 бит/16 байт).

**InvMixColumns()** – преобразование при расшифровании, которое является обратным к MixColumns()

**InvShiftRows()** – преобразование при расшифровании, которое является обратным к ShiftRows()

**InvSubBytes()** – преобразование при расшифровании, которое является обратным к SubBytes()

**K** – ключ шифрования

**MixColumns()** – преобразование при шифровании, которое перемешивает данные в каждом столбце состояния (независимо от других столбцов), чтобы получить новое значение состояния

$N_b$  – число столбцов (32-х битных слов), составляющих состояние. Для данного стандарта  $N_b = 4$  (смотри подраздел 6.3)

$N_k$  – число 32-х битных слов, составляющих ключ шифрования. Для данного стандарта  $N_k = 4, 6$  или  $8$  (смотри подраздел 6.3)

$N_r$  – число раундов.  $N_r$  является функцией  $N_k$  и  $N_b$  (которые фиксированы). Для данного стандарта  $N_r = 10, 12$  или  $14$  (смотри подраздел 6.3)

**Rcon[]** – массив слов-констант

**RotWord()** – функция, используемая в процедуре расширения ключа для циклической перестановки 4-х байтовых слов

**ShiftRows()** – преобразование при шифровании, преобразующее матрицу состояния путём циклического сдвига её трёх последних строк на различную величину

**SubBytes()** – преобразование при шифровании, которое изменяет матрицу состояния, используя нелинейную таблицу подстановки (S-блок). Каждый байт матрицы состояния обрабатывается независимо от других.

**SubWord()** – функция, используемая в процедуре расширения ключа. На вход функции поступает 4-х байтовое слово. Каждый байт слова изменяется с помощью S-блока и подаётся на выход. Таким образом, выходом функции является 4-х байтовое слово.

**XOR** – операция исключающего ИЛИ.

$\oplus$  – операция исключающего ИЛИ.

$\otimes$  – операция умножения двух многочленов (степень каждого из которых меньше 4) по модулю  $x^4 + 1$

• – операция умножения в конечном поле.

### 3. Условные знаки и соглашения

#### 3.1 Входы и выходы

Входом и выходом алгоритма AES являются последовательности 128-ми бит (цифр, которые принимают значение 0 либо 1). Эти последовательности иногда будут рассматриваться как блоки. Число бит в блоке будем называть длиной блока. Ключ шифрования для алгоритма AES – это последовательность 128, 192 или 256-ти бит. Другие длины входа, выхода и ключа шифрования в данном стандарте недопустимы.

Биты в пределах этих последовательностей будут пронумерованы, начиная от нуля и кончая числом, на единицу меньшим длины последовательности. Номер бита ( $i$ ) будем называть индексом бита. Таким образом, индекс может принадлежать одному из следующих диапазонов  $0 \leq i < 128$ ,  $0 \leq i < 192$ ,  $0 \leq i < 256$  в зависимости от длины блока и ключа (смотри выше).

#### 3.2 Байты

Основным элементом, которым оперирует алгоритм AES, является байт – последовательность восьми бит, обрабатываемых как единое целое. Последовательности, представляющие вход, выход и ключ шифрования и описанные в подразделе 3.1, обрабатываются как массивы байт. Эти массивы формируются путём разделения последовательностей бит на группы из 8-ми рядом стоящих бит так, чтобы в целом получился массив байт (смотри подраздел 3.3). Байты, составляющие вход, выход и ключ шифрования, будут обозначаться как  $a$ . Причём для указания конкретного байта будут использоваться две формы - либо  $a_n$ , либо  $a[n]$ , где  $n$  будет находиться в одном из следующих диапазонов:

длина ключа = 128 бит,  $0 \leq n < 16$ ;

длина ключа = 192 бит,  $0 \leq n < 24$ ;

длина ключа = 256 бит,  $0 \leq n < 32$ ;

длина блока = 128 бит,  $0 \leq n < 16$ ;

В алгоритме AES каждому значению байта может быть сопоставлена последовательность значений бит, составляющих этот байт. Порядок указания бит показан в фигурных скобках:  $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$ . Таким образом, используя представление в виде многочлена, байты можно интерпретировать как элементы в конечном поле:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^7 b_i x^i \quad (3.1)$$

Например, последовательность  $\{01100011\}$  определяет конкретный элемент конечного поля  $x^6 + x^5 + x + 1$ .

Указывать значение байта удобно, используя 16-ричную форму записи числа. При этом байт разделяется на две группы из 4-х бит и каждая группа битов представляется одним символом согласно рисунку 1.

биты	символ	биты	символ	биты	символ	биты	символ
0000	0	0100	4	1000	8	1100	c
0001	1	0101	5	1001	9	1101	d
0010	2	0110	6	1010	a	1110	e
0011	3	0111	7	1011	b	1111	f

Рисунок 1 – шестнадцатеричное представление последовательности бит

Таким образом, последовательность бит {01100011} можно представить 16-ричным числом {63}. Причём 16-ричный символ, представляющий старшие биты, идёт первым, а за ним следует символ, представляющий младшие биты.

При выполнении некоторых операций в конечном поле слева от последовательности 8-ми битов байта требуется ещё один дополнительный бит { $b_8$ }. Присутствие этого бита обозначается дописыванием «{01}» непосредственно перед 8-ю битами. Пример 9-ти битовой последовательности – {01}{1b}.

### 3.3 Массивы байт

Массивы байт будут представлены в следующей форме:

$$a_0 a_1 a_2 \mathbf{K} a_{15}$$

Бит во входной последовательности будем обозначать как *input*. Выделение байтов из 128-битовой входной последовательности  $input_0 input_1 input_2 \mathbf{K} input_{126} input_{127}$  происходит так:

$$a_0 = \{input_0, input_1, \mathbf{K}, input_7\}$$

$$a_1 = \{input_8, input_9, \mathbf{K}, input_{15}\}$$

•  
•  
•

$$a_{15} = \{input_{120}, input_{121}, \mathbf{K}, input_{127}\}$$

Этот пример может быть расширен на более длинные последовательности (на 192- и 256-битные ключи) согласно формуле:

$$a_n = \{input_{8n}, input_{8n+1}, \mathbf{K}, input_{8n+7}\} \quad (3.2)$$

На рисунке 2 показано, как пронумерованы биты в пределах каждого байта входной последовательности (если что-то непонятно см. подразделы 3.2 и 3.3).

номер битов входной последовательности	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
номер байта	0							1							2							...			
номер бита в байте	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Рисунок 2 – нумерация байт и бит

### 3.4 Матрица состояния

Внутри алгоритма AES выполняются операции над матрицей байт, называемой матрицей состояния или просто состоянием. Матрица состояния состоит из 4 строк, каждая из которых содержит  $N_b$  байт, где  $N_b$  - длина блока, делённая на 32. В матрице состояния, обозначаемой символом *s*, положение каждого байта определяется двумя индексами: номером строки *r* ( $0 \leq r < 4$ ), и номером столбца *c* ( $0 \leq c < N_b$ ). Это позволяет обращаться к каждому байту матрицы состояния как  $s_{r,c}$ , или как  $s[r,c]$ . Для данного стандарта  $N_b = 4$  (то есть  $0 \leq c < 4$ , см. также подраздел 6.3).

Входными данными для операций шифрования/расшифрования является массив из 16-ти байт — *in*. Перед началом шифрования/расшифрования (см. раздел 5) байты  $in_0, in_1, \dots, in_{15}$  этого массива размещаются в матрице состояния так, как показано на рисунке 3. При выполнении шифрования/расшифрования значения байтов в матрице состояния изменяются. Конечное значение матрицы состояния является выходом

алгоритма и преобразуется в последовательность выходных байт:  $out_0, out_1, \dots, out_{15}$  – массив  $out$  (см. рис.3).



Рисунок 3 – матрица состояния, входной массив и выходной массив

Таким образом, перед началом шифрования/расшифрования входные данные заносятся в матрицу состояния в соответствии с формулой:  $s[r,c]=in[r+4c]$ , где  $0 \leq r < 4$  и  $0 \leq c < N_b$  (3.3)

При выводе результата шифрования/расшифрования байты из матрицы состояния переносятся в массив  $out$  в соответствии с формулой:  $out[r+4c]=s[r,c]$ , где  $0 \leq r < 4$  и  $0 \leq c < N_b$  (3.4)

### 3.5 Матрица состояния как массив столбцов

Четыре байта в каждом столбце матрицы состояния можно рассматривать как одно 32-х битное слово. Причём номер строки матрицы состояния  $r$  является номером байта в пределах этого 4-х байтного слова. Таким образом, матрица состояния может быть представлена как одномерный массив 32-х битных слов (столбцов)  $w_0 \mathbf{K} w_3$ , где номер столбца  $c$  является индексом в этом массиве. Матрица состояния (см. рис. 3) представляется в виде массива из 4-х слов следующим образом:

$$\begin{aligned}
 w_0 &= s_{0,0} \ s_{1,0} \ s_{2,0} \ s_{3,0} \\
 w_1 &= s_{0,1} \ s_{1,1} \ s_{2,1} \ s_{3,1} \\
 w_2 &= s_{0,2} \ s_{1,2} \ s_{2,2} \ s_{3,2} \\
 w_3 &= s_{0,3} \ s_{1,3} \ s_{2,3} \ s_{3,3}
 \end{aligned}
 \tag{3.5}$$

## 4. Математическое введение

В алгоритме AES все байты рассматриваются как элементы конечного поля в соответствии с правилами, описанными в подразделе 3.2. Элементы конечного поля можно складывать и умножать. Но эти операции выполняются по правилам, отличающимся от принятых для обычных чисел. Далее будут раскрыты основные математические понятия, необходимые для понимания раздела 5.

### 4.1 Сложение

Числа в конечном поле представляются в виде многочленов. Поэтому сложение двух чисел выполняется путём сложения многочленов, представляющих эти числа. Сложение двух многочленов выполняется путём «сложения» их коэффициентов при одинаковых степенях  $x$ . «Сложение» коэффициентов выполняется с помощью операции XOR (обозначаемой как  $\oplus$ ), то есть по модулю 2 ( $1 \oplus 1=0, 1 \oplus 0=1, 0 \oplus 0=0, 0 \oplus 1=1$ ). Вычитание многочленов равнозначно сложению.

Вторым способом сложения чисел в конечном поле является суммирование по модулю 2 соответствующих бит в байте. Суммой двух байт  $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$  и  $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$  будет байт  $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$ , где  $c_i = a_i \oplus b_i$  (то есть  $c_7 = a_7 \oplus b_7, c_6 = a_6 \oplus b_6, \mathbf{K}$ ,

$$c_0 = a_0 \oplus b_0).$$

Следующие три записи эквивалентны друг другу:

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 \quad (\text{представление в виде многочленов})$$

$$\{01010111\} \oplus \{10000011\} = \{11010100\} \quad (\text{двоичное представление})$$

$$\{57\} \oplus \{83\} = \{d4\} \quad (16\text{-ричное представление})$$

## 4.2 Умножение

При представлении чисел в виде многочленов, умножение в поле  $GF(2^8)$  (обозначаемое как « $\bullet$ ») выполняется в 2 этапа. Сначала многочлены-множители перемножаются по обычным правилам (при этом операция сложения выполняется по правилам поля, см. подраздел 4.1). Затем результат умножения приводится к модулю, задаваемому неприводимым многочленом степени 8. Получившееся число будет результатом умножения многочленов в поле.

Многочлен называется неприводимым, если он делится нацело только на единицу и на самого себя. Для алгоритма AES этим неприводимым многочленом является

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (4.1)$$

или  $\{01\}\{1b\}$  в 16-ричном представлении.

Например  $\{57\} \bullet \{83\} = \{c1\}$ .

Подробнее:

$$\begin{aligned} (x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad + x^7 + x^5 + x^3 + x^2 + x + \\ &\quad + x^6 + x^4 + x^2 + x + 1 = \end{aligned}$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

Не забывайте, что сложение многочленов выполняется по правилам поля (см. подраздел 4.1).

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

$$\text{Итого: } (x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) = x^7 + x^6 + 1.$$

Приведение к модулю  $m(x)$  означает, что результатом умножения двух многочленов в поле будет многочлен со степенью меньше 8-ми. И этот многочлен можно представить как байт. В отличие от сложения, нет простой операции над байтами, которая соответствовала бы умножению.

Операция умножения, определённая выше, является ассоциативной, и элемент  $\{01\}$  является единицей поля. Для любого ненулевого бинарного многочлена  $b(x)$  со степенью меньше 8-ми может быть найден обратный к нему по умножению многочлен, обозначаемый как  $b^{-1}(x)$ . Для нахождения обратного элемента используют расширенный алгоритм Евклида [7], с помощью которого вычисляют многочлены  $a(x)$  и  $c(x)$ , такие что

$$b(x)a(x) + m(x)c(x) = 1 \quad (4.2)$$

Следовательно,  $a(x) \cdot b(x) \bmod m(x) = 1$ , что означает

$$b^{-1}(x) = a(x) \bmod m(x) \quad (4.3)$$

Кроме того, для любых  $a(x)$ ,  $b(x)$  и  $c(x)$ , принадлежащих полю, выполняется

$$a(x) \cdot (b(x) + c(x)) = a(x) \cdot b(x) + a(x) \cdot c(x)$$

Отсюда следует, что множество 256-ти возможных числовых значений байта, операция XOR, используемая в качестве сложения, и операция умножения, определённая выше, образуют конечное поле  $GF(2^8)$ .

#### 4.2.1 Умножение на $x$

Произведение бинарного многочлена, определённого выражением (3.1), на многочлен  $x$  даёт многочлен

$$b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x \quad (4.4)$$

Результат умножения  $x \cdot b(x)$  получается приведением полученного многочлена (4.4) к модулю  $m(x)$ , который определён выражением (4.1). Если  $b_7 = 0$ , то выражению (4.4) приведение к модулю  $m(x)$  не требуется. Если  $b_7 = 1$ , то приведение к модулю  $m(x)$  выполняется путём вычитания (операцией XOR) многочлена  $m(x)$ . Это означает, что умножение на  $x$  (то есть на  $\{00000010\}$  или  $\{02\}$ ) может быть выполнено на уровне байт. Сначала выполняется сдвиг влево, а затем (если требуется) выполняется побитовое сложение операций XOR с числом  $\{1b\}$ . Эта операция над байтами обозначается как `xtime()`. Умножение на более высокие степени  $x$  может быть выполнено путём повторения применения операции `xtime()`. Таким образом, умножение числа на любую константу может быть выполнено путём умножения на нужные степени  $x$  и сложения промежуточных результатов.

Например  $\{57\} \bullet \{13\} = \{fe\}$

Подробнее:

$$\{57\} \bullet \{02\} = \text{xtime}(\{57\}) = \{ae\}$$

$$\{57\} \bullet \{04\} = \text{xtime}(\{ae\}) = \{47\}$$

$$\{57\} \bullet \{08\} = \text{xtime}(\{47\}) = \{8e\}$$

$$\{57\} \bullet \{10\} = \text{xtime}(\{8e\}) = \{07\}$$

$$\begin{aligned} \text{Итак: } \{57\} \bullet \{13\} &= \{57\} \bullet (\{01\} \oplus \{02\} \oplus \{10\}) = \\ &= \{57\} \oplus \{ae\} \oplus \{07\} = \\ &= \{fe\} \end{aligned}$$

#### 4.3 Многочлены с коэффициентами, принадлежащими полю $\text{GF}(2^8)$

Многочлены 4-й степени с коэффициентами, являющимися элементами конечного поля, определяются с помощью выражения:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \quad (4.5)$$

Выражение (4.5) далее будет представляться в форме слова как  $[a_0, a_1, a_2, a_3]$ . Заметьте, что выполнение операций над этими многочленами несколько отличается от аналогичных действий над многочленами, использовавшимися при определении элементов конечного поля, хотя оба типа многочленов используют одну и ту же независимую переменную  $x$ .

В данном подразделе стандарта коэффициенты многочленов являются элементами конечного поля (то есть используются байты вместо битов). Кроме того, при умножении многочленов 4-й степени используется другое приведение к модулю, определённое ниже. Различия будут ясны из контекста.

Для иллюстрации сложения и умножения определим ещё один многочлен 4-й степени:

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0 \quad (4.6)$$

Сложение многочленов выполняется путём сложения их коэффициентов при одинаковых степенях  $x$ . При этом сложение коэффициентов производится по правилам, определённым для элементов конечного поля. Таким образом, сложение многочленов выполняется с помощью операции XOR над соответствующими байтами каждого слова, представляющего многочлен. То есть выполняется операция XOR над двумя словами.

Используя выражения (4.5) и (4.6), получаем:

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0) \quad (4.7)$$

Умножение выполняется в два шага. На первом шаге вычисления произведения  $c(x) = a(x) \bullet b(x)$  происходит алгебраическое раскрытие скобок и группировка коэффициентов при одинаковых степенях  $x$ . Получается многочлен:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0 \quad (4.8)$$

где

$$\begin{aligned} c_0 &= a_0 \bullet b_0 \\ c_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 \\ c_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \\ c_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3 \\ c_4 &= a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\ c_5 &= a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\ c_6 &= a_3 \bullet b_3 \end{aligned} \quad (4.9)$$

Многочлен  $c(x)$  нельзя представить в виде 4-х байтового слова. Поэтому вторым шагом является приведение многочлена  $c(x)$  к модулю, задаваемому многочленом степени 4. Результатом будет многочлен, степень которого меньше 4.

**В алгоритме AES в качестве модуля используется многочлен  $x^4 + 1$ .** При этом выполняется:

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4} \quad (4.10)$$

Произведением многочленов  $a(x)$  и  $b(x)$  (обозначаемым как  $a(x) \otimes b(x)$ ) является многочлен

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0 \quad (4.11)$$

где

$$\begin{aligned} d_0 &= (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\ d_1 &= (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\ d_2 &= (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \\ d_3 &= (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3) \end{aligned} \quad (4.12)$$

Если  $a(x)$  – фиксированный многочлен, то операцию умножения двух многочленов можно представить в матричной форме:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (4.13)$$

Так как  $x^4 + 1$  не является неприводимым многочленом в поле  $\text{GF}(2^8)$ , то умножение на фиксированный 4-х членный многочлен не всегда обратимо. Тем не менее, в алгоритме AES используется фиксированный 4-х членный многочлен, который имеет обратный (см. пункты 5.1.3 и 5.3.3):

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (4.14)$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (4.15)$$

Другой многочлен, используемый в алгоритме AES (смотри функцию RotWord() в подразделе 5.2), имеет значения коэффициентов:  $a_0 = a_1 = a_2 = \{00\}$ ,  $a_3 = \{01\}$  – что соответствует многочлену  $x^3$ . Из выражения (4.13) видно, что результатом умножения входного слова на фиксированный многочлен  $x^3$  будет выходное слово, полученное перемешиванием байтов входного слова. То есть слово  $[b_0, b_1, b_2, b_3]$ , будет преобразовано в слово  $[b_1, b_2, b_3, b_0]$ .

## 5. Описание алгоритма

В алгоритме AES размер **входного блока, выходного блока и матрицы состояния равен 128-ми битам**. Это достигается путём задания  $N_b = 4$ , то есть матрица состояния имеет 4 столбца.

В алгоритме AES **длина ключа шифрования К равна 128, 192 или 256 бит**. Длина ключа задаётся с помощью переменной  $N_K$  равной 4, 6 или 8 (количество 32-х разрядных слов в ключе шифрования).

Количество раундов алгоритма AES зависит от длины ключа. Количество раундов задаётся переменной  $N_r$ , где  $N_r = 10$  при  $N_K = 4$ ,  $N_r = 12$  при  $N_K = 6$  и  $N_r = 14$  при  $N_K = 8$ .

**В данном стандарте используются только те комбинации (длина ключа) - (размер блока) - (число раундов), которые представлены на рисунке 4.** Для уточнения вопросов, связанных с длиной ключа, размером блока и количеством раундов, смотри подраздел 6.3.

При шифровании и расшифровании в алгоритме AES используется функция раунда, состоящая из 4-х отдельных байт-ориентированных преобразований: 1) замена байт с помощью таблицы подстановки (SubBytes), 2) сдвиг строк матрицы состояния на различную величину (ShiftRows), 3) перемешивание данных в пределах каждого столбца матрицы состояния (MixColumns), и 4) сложение ключа раунда с матрицей состояния (AddRoundKey). Эти преобразования (и обратные к ним) описаны в пунктах 5.1.1 – 5.1.4 и 5.3.1 – 5.3.4. Процедура шифрования и расшифрования описаны в подразделах 5.1 и 5.3 соответственно. Процедура получения ключей раундов описана в подразделе 5.2.

	Длина ключа ( $N_K$ слов)	Размер блока ( $N_b$ слов)	Число раундов ( $N_r$ )
<b>AES-128</b>	<b>4</b>	<b>4</b>	<b>10</b>
<b>AES-192</b>	<b>6</b>	<b>4</b>	<b>12</b>
<b>AES-256</b>	<b>8</b>	<b>4</b>	<b>14</b>

Рисунок 4 – комбинации длины ключа, размера блока и числа раундов

### 5.1 Шифрование (процедура Cipher)

Сначала входные данные копируются в матрицу состояния по правилам, описанным в подразделе 3.4. Затем производится сложение матрицы состояния и ключа раунда. После этого матрица состояния преобразуется с помощью функции раунда 10, 12 или 14 раз (в зависимости от длины ключа). Причём последний раунд обработки немного отличается от предыдущих  $N_r - 1$  раундов. В конце матрица состояния копируется в массив выходных данных по правилам, описанным в подразделе 3.4.

Функция раунда использует параметр – массив подключей  $w[]$  (ключей раундов). Этот массив представляет собой массив 4-х байтовых слов, который получается с помощью процедуры расширения ключа (см. подраздел 5.2).

Процедура шифрования представлена в псевдокоде на рисунке 5. Отдельные преобразования – SubBytes, ShiftRows, MixColumns и AddRoundKey – обрабатывают матрицу состояния. Они описаны далее.

```

in[ i ] - массив байт (вход),  $0 \leq i \leq 15$ 
out[ i ] - массив байт (выход),  $0 \leq i \leq 15$ 
w[ i ] - массив слов (массив подключей),  $0 \leq i \leq N_b \cdot (N_r + 1) - 1$ 

Cipher ( in, out, w )
  начало
  создаём переменную state — матрицу состояния
  создаём переменную round — номер раунда

  state := in
  AddRoundKey( state, w, 0)           // см. пункт 5.1.4
  для round от 1 до  $N_r - 1$  с шагом 1
    SubBytes( state)                 // см. пункт 5.1.1
    ShiftRows( state)                // см. пункт 5.1.2
    MixColumns( state)               // см. пункт 5.1.3
    AddRoundKey( state, w, round )
  всё-цикл

  SubBytes( state)                   // см. пункт 5.1.1
  ShiftRows( state)                  // см. пункт 5.1.2
  AddRoundKey( state,  $N_r$  )

  out := state
  конец

```

Рисунок 5 – описание шифрования с помощью псевдокода

Как видно из рисунка 5, все  $N_r$  раундов одинаковы, кроме последнего, в котором отсутствует процедура MixColumns.

В приложении Б представлен пример выполнения шифрования. Показано значение матрицы состояния перед началом каждого раунда, а также после выполнения каждого из 4-х преобразований, описанных далее.

### 5.1.1 Процедура SubBytes

Процедура SubBytes выполняет нелинейную замену байтов в матрице состояния с помощью таблицы подстановки (S-блока). При этом каждый байт обрабатывается независимо от других.

Таблица подстановки имеет обратную и составлена путём выполнения двух действий:

1. нахождение обратного по умножению элемента в поле  $GF(2^8)$  как это описано в подразделе 4.2 (элемент {00} переходит сам в себя)
2. применение следующего аффинного преобразования ( над  $GF(2)$  ):

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \quad (5.1)$$

где  $0 \leq i < 8$ ,

$b_i$  -  $i$ -й бит байта,

$c_i$  -  $i$ -й бит байта  $c$ , который имеет значение {63} или {01100011}.

Здесь и далее штрих над переменной (например  $b'$ ) будет означать, что переменная получила более новое значение. В матричной форме аффинное преобразование элементов таблицы подстановки может быть представлено так:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (5.2)$$

На рисунке 6 показана обработка матрицы состояния с помощью таблицы подстановки.

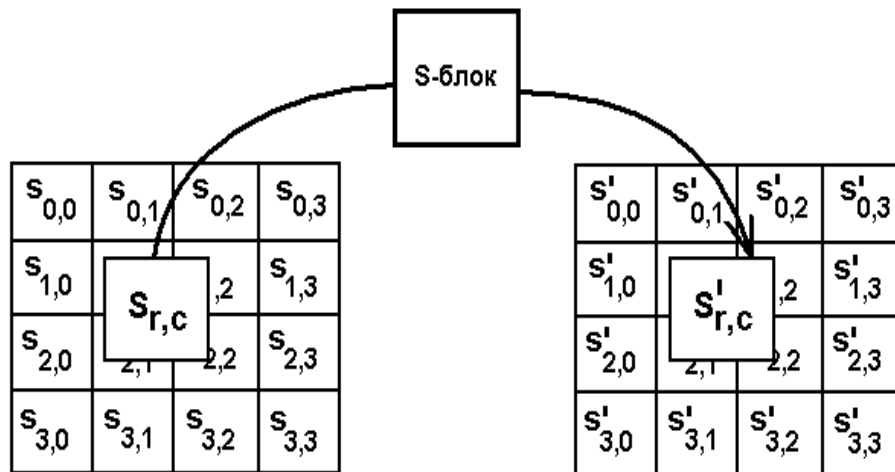


Рисунок 6 – замена каждого байта матрицы состояния с помощью таблицы подстановки

Таблица подстановки, используемая процедурой SubBytes, представлена в 16-ричном виде на рисунке 7. Например, если  $s_{1,1} = \{53\}$ , то результат замены следует искать на пересечении строки с индексом “5” и столбца с индексом “3” (см. рис 7). То есть  $s'_{1,1} = \{ed\}$ .

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

Рисунок 7 – значения таблицы подстановки для байта {xy} (в 16-ричном формате)

### 5.1.2 Процедура ShiftRows

Процедура ShiftRows выполняет циклический сдвиг байтов в последних трёх строках матрицы состояния на различное число байт. Первая строка,  $r = 0$ , не сдвигается.

Точно процедуру можно описать так:

$$s'_{r,c} = s_{r,(c + \text{shift}(r, N_b)) \bmod N_b} \quad \text{для } 0 < r < 4 \text{ и } 0 \leq c < N_b \quad (5.3)$$

где величина сдвига  $\text{shift}(r, N_b)$  зависит от номера строки  $r$  следующим образом (напомним, что  $N_b = 4$ ):

$$\text{shift}(1,4)=1 ; \text{shift}(2,4)=2 ; \text{shift}(3,4)=3. \quad (5.4)$$

В результате байты сдвигаются на младшие позиции в строке (то есть на позиции с меньшим значением индекса  $c$ ), в то время как байты, бывшие в самых младших позициях, перемещаются в старшие позиции (то есть на позиции с большим значением индекса  $c$ ). Рисунок 8 поясняет выполнение процедуры ShiftRows.

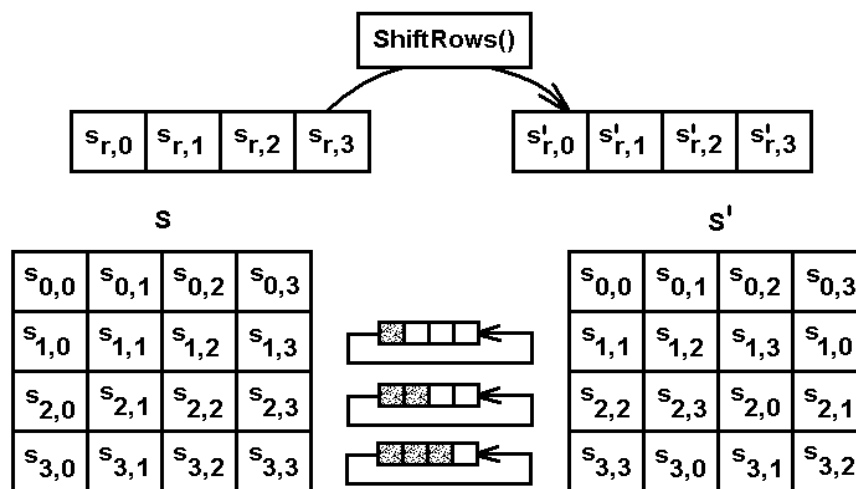


Рисунок 8 – процедура ShiftRows циклически сдвигает байты в трёх последних строках матрицы состояния

### 5.1.3 Процедура MixColumns

Процедура MixColumns обрабатывает матрицу состояния столбец за столбцом, рассматривая каждый столбец как 4-х членный многочлен, как это описано в подразделе 4.3. Столбцы рассматриваются как многочлены в поле  $GF(2^8)$  и умножаются по модулю  $x^4 + 1$  на фиксированный многочлен  $a(x)$ :

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}. \quad (5.5)$$

Это можно записать в матричной форме (см. подраздел 4.3):

$$s'(x) = a(x) \otimes s(x) : \begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{для } 0 \leq c < N_b \quad (5.6)$$

В итоге байты столбца изменяются в соответствии со следующими выражениями:

$$\begin{aligned} s'_{0,c} &= (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c} \\ s'_{1,c} &= s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c} \\ s'_{2,c} &= s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}) \end{aligned}$$

Рисунок 9 иллюстрирует работу процедуры MixColumns.

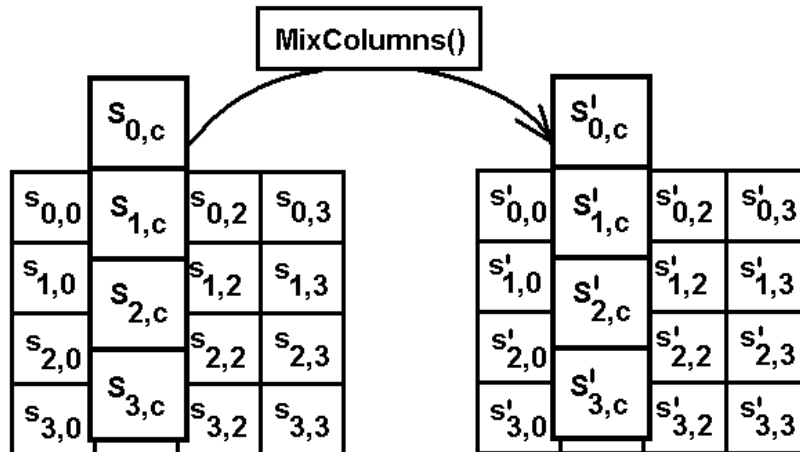


Рисунок 9 – процедура MixColumns обрабатывает матрицу состояния столбец за столбцом.

#### 5.1.4 Процедура AddRoundKey

Процедура AddRoundKey выполняет сложение ключа раунда и матрицы состояния с помощью побитовой операции XOR. В качестве параметров процедура получает матрицу состояния, массив подключей ( $w[]$ ) и номер раунда ( $round$ ). Каждый ключ раунда состоит из  $N_b$  слов. Каждое из указанных  $N_b$  слов складывается с соответствующим столбцом матрицы состояния согласно выражению:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \oplus w[round * N_b + c] \quad (5.7)$$

где  $0 \leq c < N_b$ ,

$w$  - массив подключей, описанный в подразделе 5.2,

$round$  – номер раунда ( $0 \leq round \leq N_r$ ).

При шифровании первое сложение ключа раунда происходит при  $round=0$ , до первого выполнения функции раунда (см. рис.5). Выполнение процедуры AddRoundKey в течение остальных  $N_r$  раундов происходит при  $1 \leq round \leq N_r$ .

Выполнение процедуры показано на рисунке 10, где  $l = round * N_b$ . Адресация байтов в пределах слова описана в подразделе 3.1.

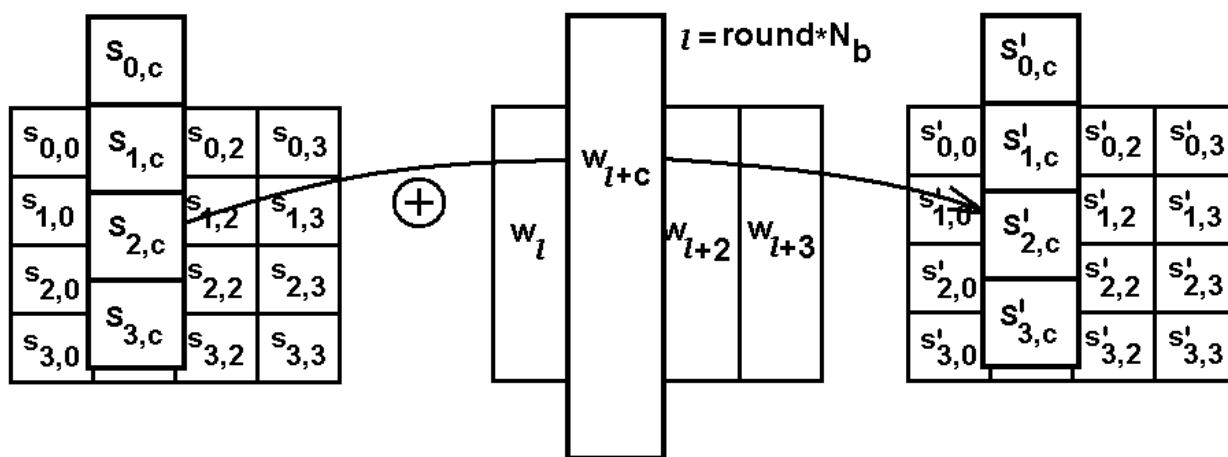


Рисунок 10 – процедура AddRoundKey выполняет операцию XOR над каждым столбцом матрицы состояния и словом из массива подключей.

## 5.2 Получение подключей (процедура KeyExpansion)

Процедура KeyExpansion используется в алгоритме AES для генерации подключей (ключей раундов) из ключа шифрования  $K$ . Ещё эту процедуру называют расширением ключа. Всего генерируется  $N_b(N_r + 1)$  слов: в начале алгоритма требуется  $N_b$  слов и затем в каждом из  $N_r$  раундов требуется  $N_b$  слов ключевых данных. В результате массив подключей представляет собой массив 4-х байтовых слов, обозначаемых как  $[w_i]$ , где  $i$  находится в пределах  $0 \leq i < N_b(N_r + 1)$ .

Процесс преобразования входного ключа в массив подключей показан с помощью псевдокода на рисунке 11.

Функция SubWord() принимает на вход 4-х байтовое входное слово и преобразует его в выходное слово. Выходное слово образуется путём замены каждого байта входного слова с помощью S-блока (см. пункт 5.1.1, рис. 7).

Функция RotWord() принимает на вход слово  $[a_0, a_1, a_2, a_3]$  и выполняет циклическую перестановку, возвращая слово  $[a_1, a_2, a_3, a_0]$ .

Rcon[] – массив слов-констант. Значение слова Rcon[i] определяется согласно выражению:  $[x^{i-1}, \{00\}, \{00\}, \{00\}]$ , где  $x^{i-1}$  – степень  $x$ , записанная по правилам поля  $GF(2^8)$ . То есть  $x$  обозначается как  $\{02\}$  (см. подраздел 4.2). Заметим, что начальным значением индекса  $i$  в массиве Rcon[] является 1, а не 0.

Из рисунка 11 видно, что в первые  $N_k$  слов массива подключей копируется ключ шифрования. Каждое последующее слово,  $w[i]$ , получается путём выполнения операции XOR над предыдущим словом,  $w[i-1]$ , и словом, находящимся на  $N_k$  позиций раньше,  $w[i - N_k]$ .

При вычислении слов, находящихся на позициях, кратных  $N_k$ , над словом  $w[i-1]$  производятся дополнительные операции. Слово  $w[i-1]$  обрабатывается функцией RotWord(). Затем получившееся слово идёт на вход функции SubWord(). Затем результат работы функции SubWord() складывается с помощью операции XOR со словом из массива констант Rcon[].

Важно заметить, что вычисление подключей для 256-битного ключа ( $N_k = 8$ ) немного отличается от аналогичной операции для 128- и 192-битных ключей. Если  $N_k = 8$  и  $i-4$  кратно  $N_k$ , то перед выполнением операции XOR слово  $w[i-1]$  обрабатывается функцией SubWord().

```

K[i] - массив байт (ключ шифрования),  $0 \leq i \leq 4 \cdot N_K - 1$ 
w[i] - массив слов (массив подключей),  $0 \leq i \leq N_B \cdot (N_r + 1) - 1$ 

KeyExpansion (K, w)
  начало
    создаём локальную переменную - слово temp
    создаём локальную переменную - i - счётчик слов
    i := 0
    пока ( i < NK )
      w[i] := слово ( K[4*i], K[4*i+1], K[4*i+2], K[4*i+3] )
      i := i+1
    конец-цикл-пока

    i := NK
    пока ( i < NB * (Nr+1) )
      temp := w[ i - 1 ]
      если ( i mod NK = 0 )
        temp := SubWord(RotWord(temp)) xor Rcon[ ( i div NK ) ]
      иначе
        если ( NK > 6 И ( i mod NK = 4 ) ), то temp := SubWord(temp), конец-если
      конец-если
      w[ i ] := w[ i - NK ] xor temp
      i := i+1
    конец-цикл-пока

  конец

```

Для краткости был построен единый алгоритм для всех трёх возможных значений переменной  $N_K$ . Требования к длине ключа представлены в подразделе 6.1.

Рисунок 11 – псевдокод процедуры KeyExpansion

В приложении А представлен пример выполнения процедуры KeyExpansion.

### 5.3 Расшифрование (процедура InvCipher)

Преобразования, составляющие процедуру шифрования, могут быть инвертированы и применены в обратном порядке для получения процедуры расшифрования алгоритма AES – InvCipher. В процедуре расшифрования используются следующие отдельные преобразования: InvShiftRows, InvSubBytes, InvMixColumns и AddRoundKey. Эти преобразования обрабатывают матрицу состояния и описаны далее. Описание процедуры расшифрования с помощью псевдокода представлено на рисунке 12. Массив w[] на рисунке 12 содержит подключи и описан в подразделе 5.2.

```

in [ i ] - массив байт (вход),  $0 \leq i \leq 15$ 
out [ i ] - массив байт (выход),  $0 \leq i \leq 15$ 
w [ i ] - массив слов (массив подключей),  $0 \leq i \leq N_b * (N_r + 1) - 1$ 

InvChipher( in, out, w )
начало

    создаём переменную state – матрицу состояния
    создаём переменную round – номер раунда

    state := in

    AddRoundKey( state, w, Nr )           // см. пункт 5.1.4

    для round от Nr-1 до 1 с шагом -1
        InvShiftRows( state )           // см. пункт 5.3.1
        InvSubBytes( state )           // см. пункт 5.3.2
        AddRoundKey( state, w, round )
        InvMixColumns( state )         // см. пункт 5.3.3
    всё - цикл

    InvShiftRows( state )
    InvSubBytes( state )
    AddRoundKey( state, w, 0 )

    out := state

конец

```

Рисунок 12 – описание процедуры расшифрования с помощью псевдокода

### 5.3.1 Процедура **InvShiftRows**

Процедура **InvShiftRows** выполняет действия, обратные процедуре **ShiftRows**. Байты в последних трёх строках матрицы состояния циклически сдвигаются на различное число байт. Первая строка с номером  $r = 0$  не сдвигается. Нижние три строки циклически сдвигаются на  $N_b - shift(r, N_b)$  байт, где величина сдвига  $shift(r, N_b)$  зависит от номера строки и определена выражением 5.4 (см. пункт 5.1.2).

Точно процедуру **InvShiftRows()** можно описать так:

$$s'_{r, (c + shift(r, N_b)) \bmod N_b} = s_{r, c}, \text{ где } 0 < r < 4 \text{ и } 0 \leq c < N_b \quad (5.8)$$

Рисунок 13 иллюстрирует процедуру InvShiftRows.

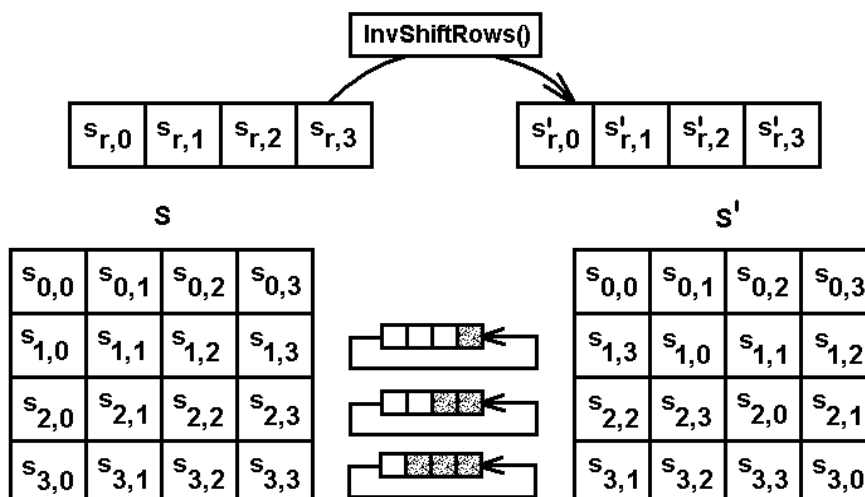


Рисунок 13 – процедура InvShiftRows циклически сдвигает байты в трёх последних строках матрицы состояния.

### 5.3.2 Процедура InvSubBytes

Процедура InvSubBytes является инверсией процедуры SubBytes. Процедура InvSubBytes выполняет замену каждого байта матрицы состояния в соответствии с обратной таблицей замены (инвертированным S-блоком). Для этого необходимо найти обратный по умножению элемент в поле  $GF(2^8)$  и затем применить обратное аффинное преобразование. Инвертированный S-блок, используемый в процедуре InvSubBytes, показан на рисунке 14.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

Рисунок 14 – значения обратной таблицы подстановки для байта {xy} (в 16-ричном формате)

### 5.3.3 Процедура InvMixColumns

Процедура InvMixColumns является обратной к процедуре MixColumns. Процедура InvMixColumns обрабатывает матрицу состояния столбец за столбцом, рассматривая каждый столбец как 4-х членный многочлен, как это описано в подразделе 4.3. Столбцы

рассматриваются как многочлены в поле  $GF(2^8)$  и умножаются по модулю  $x^4 + 1$  на фиксированный многочлен  $a^{-1}()$ :

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (5.9)$$

Это можно записать в матричной форме (см. подраздел 4.3):

$$s'(x) = a^{-1}(x) \otimes s(x) : \begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \text{ для } 0 \leq c < N_b \quad (5.10)$$

В итоге байты столбца изменяются в соответствии со следующими выражениями:

$$\begin{aligned} s'_{0,c} &= (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c}) \\ s'_{1,c} &= (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c}) \\ s'_{2,c} &= (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c}) \\ s'_{3,c} &= (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c}) \end{aligned}$$

### 5.3.4 Инверсия процедуры AddRoundKey

Процедура AddRoundKey, описанная в пункте 5.1.4, является обратной сама к себе, так как состоит только в применении операции XOR.

### 5.3.5 Эквивалентная процедура расшифрования

В процедуре расшифрования, описанной в подразделе 5.3 и показанной на рисунке 12, порядок применения преобразований отличается от порядка, применённого в процедуре шифрования. При этом для шифрования и расшифрования применяется один и тот же массив подключей. Однако, некоторые свойства алгоритма AES позволяют использовать эквивалентную процедуру расшифрования, в которой последовательность преобразований совпадает с применённой в процедуре шифрования (при этом преобразования заменяются на их инверсии). Это достигается путём изменения массива подключей.

Два свойства, определяющие существование эквивалентной процедуры расшифрования – это:

1 – процедуры SubBytes и ShiftRows коммутативны. То есть, результат выполнения процедуры SubBytes непосредственно после процедуры ShiftRows будет эквивалентен результату выполнения процедуры ShiftRows непосредственно после процедуры SubBytes. То же верно и для инверсий этих процедур – InvSubBytes и InvShiftRows.

2 – операции перемешивания в столбце – MixColumns и InvMixColumns – являются линейными по отношению к данным, расположенным в столбце, что означает  $InvMixColumns(state \text{ XOR Round Key}) =$

$$= InvMixColumns(state) \text{ XOR } InvMixColumns(Round Key)$$

Согласно этим свойствам порядок выполнения процедур InvSubBytes и InvShiftRows можно изменять на обратный. Порядок выполнения процедур AddRoundKey и InvMixColumns тоже можно инвертировать, если при этом слова в последовательности подключей будут изменены с помощью процедуры InvMixColumns.

Эквивалентная процедура расшифрования получается путём инвертирования порядка выполнения процедур InvSubBytes и InvShiftRows и инвертирования порядка выполнения процедур AddRoundKey и InvMixColumns. При этом ко всем подключам кроме первого и

последнего применяется процедура `InvMixColumns`. То есть не изменяются только первые и последние  $N_b$  слов массива подключей.

С учётом этих изменений структура эквивалентной процедуры расшифрования получается более эффективной, чем у процедуры, описанной в подразделе 5.3 и показанной на рисунке 12. Псевдокод эквивалентной процедуры расшифрования представлен на рис. 15. Массив слов `dw[]` содержит модифицированные подключи. Необходимые изменения процедуры `KeyExpansion` также представлены на рис. 15.

$in[i]$  - массив байт (вход),  $0 \leq i \leq 15$   
 $out[i]$  - массив байт (выход),  $0 \leq i \leq 15$   
 $dw[i]$  - массив слов (массив подключей),  $0 \leq i \leq (N_r + 1) * N_b - 1$

**EqInvCipher( in, out, dw)**

начало

создаём переменную state – матрицу состояния

создаём переменную round – номер раунда

state := in

AddRoundKey(state, dw,  $N_r$ )

для round от  $N_r - 1$  до 1 с шагом -1

  InvSubBytes(state)

  InvShiftRows(state)

  InvMixColumns(state)

  AddRoundKey(state, dw, round)

всё-цикл

  InvSubBytes(state)

  InvShiftRows(state)

  AddRoundKey(state, dw, 0)

  out := state

конец

Для корректной работы эквивалентной процедуры расшифрования необходимо в конце процедуры KeyExpansion добавить следующий псевдокод (см. подраздел 5.2):

для  $i$  от 0 до  $((N_r + 1) * N_b - 1)$  с шагом 1

$dw[i] := w[i]$

всё-цикл

для round от 1 до  $N_r - 1$  с шагом 1

  InvMixColumns(  $dw[round * N_b]$ ,  $dw[round * N_b + 1]$ , ...,  $dw[round * N_b + N_b - 1]$  )

  // необходима смена типа переменной

всё-цикл

Заметим, что процедура InvMixColumns принимает на вход матрицу состояния, то есть двухмерный массив байт, в то время как в данном случае на её вход поступает ключ раунда, то есть одномерный массив слов. Поэтому для корректной работы процедуры необходима смена типа.

Рисунок 15 – псевдокод эквивалентной процедуры расшифрования.

## **6. Вопросы реализации**

### **6.1 Требования к длине ключа**

Реализация алгоритма AES должна поддерживать как минимум одну из трёх длин ключей, описанных в разделе 5: 128, 192 и 256 бит ( $N_k = 4, 6$  и  $8$  соответственно).

Реализации алгоритма могут опционально поддерживать две или три длины ключа, что будет способствовать совместимости различных реализаций.

### **6.2 Ограничения на выбор ключа**

Для алгоритма AES не было обнаружено слабых и полуслабых ключей. Ограничения на выбор ключа отсутствуют.

### **6.3 Выбор длины ключа, размера блока и числа раундов**

Данный стандарт однозначно определяет допустимые значения длины ключа ( $N_k$ ), размера блока ( $N_b$ ) и числа раундов ( $N_r$ ) – смотри рисунок 4. Однако в дальнейших версиях этого стандарта предполагается изменять разрешённые значения этих параметров. Разработчикам рекомендуется учитывать это обстоятельство.

### **6.4 Рекомендации по реализации алгоритма на различных платформах**

Возможны различные варианты реализации алгоритма, что во многих случаях приводит к повышению производительности или получению других преимуществ. Любая реализация, преобразующая полученный входной ключ и данные (открытый текст или шифртекст) в выходные данные (шифртекст или открытый текст) в соответствии с описанным в данном стандарте алгоритмом, является приемлемой реализацией алгоритма AES.

Рекомендации по рациональной реализации алгоритма AES на различных платформах приведены в книге [3] и других документах, доступных по ссылке [1].

## Приложение А – пример получения подключей.

В данном приложении показан ход выполнения процедуры получения подключей (KeyExpansion) для различных длин ключей. Заметим, что многобайтные числа представлены по правилам, описанным в разделе 3. Промежуточные результаты, получаемые в процессе выполнения процедуры, представлены ниже в таблице (все значения представлены в 16-ричном формате кроме номера столбца (i)).

### А.1 Расширение 128-ми битного ключа.

Здесь показан процесс получения подключей из следующего ключа шифрования:

$K = 2b\ 7e\ 15\ 16\ 28\ ae\ d2\ a6\ ab\ f7\ 15\ 88\ 09\ cf\ 4f\ 3c.$

$N_k = 4$ , следовательно

$w_0 = 2b7e1516, w_1 = 28aed2a6, w_2 = abf71588, w_3 = 09cf4f3c$

i (дес)	temp	после RotWord()	после SubWord()	Rcon[i/Nk]	после XOR с Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafa17
5	a0fafa17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafa17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d
13	3d80477d					7a96b943	4716fe3e
14	4716fe3e					5935807a	1e237e44
15	1e237e44					7359f67f	6d7a883b
16	6d7a883b	7a883b6d	dac4e23c	08000000	d2c4e23c	3d80477d	ef44a541
17	ef44a541					4716fe3e	a8525b7f
18	a8525b7f					1e237e44	b671253b
19	b671253b					6d7a883b	db0bad00
20	db0bad00	0bad00db	2b9563b9	10000000	3b9563b9	ef44a541	d4d1c6f8
21	d4d1c6f8					a8525b7f	7c839d87
22	7c839d87					b671253b	caf2b8bc
23	caf2b8bc					db0bad00	11f915bc

24	11f915bc	f915bc11	99596582	20000000	b9596582	d4d1c6f8	6d88a37a
25	6d88a37a					7c839d87	110b3efd
26	110b3efd					caf2b8bc	dbf98641
27	dbf98641					11f915bc	ca0093fd
28	ca0093fd	0093fdca	63dc5474	40000000	23dc5474	6d88a37a	4e54f70e
29	4e54f70e					110b3efd	5f5fc9f3
30	5f5fc9f3					dbf98641	84a64fb2
31	84a64fb2					ca0093fd	4ea6dc4f
32	4ea6dc4f	a6dc4f4e	2486842f	80000000	a486842f	4e54f70e	ead27321
33	ead27321					5f5fc9f3	b58dbad2
34	b58dbad2					84a64fb2	312bf560
35	312bf560					4ea6dc4f	7f8d292f
36	7f8d292f	8d292f7f	5da515d2	1b000000	46a515d2	ead27321	ac7766f3
37	ac7766f3					b58dbad2	19fadc21
38	19fadc21					312bf560	28d12941
39	28d12941					7f8d292f	575c006e
40	575c006e	5c006e57	4a639f5b	36000000	7c639f5b	ac7766f3	d014f9a8
41	d014f9a8					19fadc21	c9ee2589
42	c9ee2589					28d12941	e13f0cc8
43	e13f0cc8					575c006e	b6630ca6

## A.2 Расширение 192-битного ключа

Здесь показан процесс получения подключей из следующего ключа шифрования:

K = 8e 73 b0 f7 da 0e 64 52 c8 10 f3 2b  
80 90 79 e5 62 f8 ea d2 52 2c 6b 7b

$N_K = 6$ , следовательно

$w_0 = 8e73b0f7$ ,  $w_1 = da0e6452$ ,  $w_2 = c810f32b$ ,  $w_3 = 809079e5$   
 $w_4 = 62f8ead2$ ,  $w_5 = 522c6b7b$

i (дек)	temp	после RotWord()	после SubWord()	Rcon[i/Nk]	после XOR с Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
6	522c6b7b	2c6b7b52	717f2100	01000000	707f2100	8e73b0f7	fe0c91f7
7	fe0c91f7					da0e6452	2402f5a5
8	2402f5a5					c810f32b	ec12068e

9	ec12068e					809079e5	6c827f6b
10	6c827f6b					62f8ead2	0e7a95b9
11	0e7a95b9					522c6b7b	5c56fec2
12	5c56fec2	56fec25c	b1bb254a	02000000	b3bb254a	fe0c91f7	4db7b4bd
13	4db7b4bd					2402f5a5	69b54118
14	69b54118					ec12068e	85a74796
15	85a74796					6c827f6b	e92538fd
16	e92538fd					0e7a95b9	e75fad44
17	e75fad44					5c56fec2	bb095386
18	bb095386	095386bb	01ed44ea	04000000	05ed44ea	4db7b4bd	485af057
19	485af057					69b54118	21efb14f
20	21efb14f					85a74796	a448f6d9
21	a448f6d9					e92538fd	4d6dce24
22	4d6dce24					e75fad44	aa326360
23	aa326360					bb095386	113b30e6
24	113b30e6	3b30e611	e2048e82	08000000	ea048e82	485af057	a25e7ed5
25	a25e7ed5					21efb14f	83b1cf9a
26	83b1cf9a					a448f6d9	27f93943
27	27f93943					4d6dce24	6a94f767
28	6a94f767					aa326360	c0a69407
29	c0a69407					113b30e6	d19da4e1
30	d19da4e1	9da4e1d1	5e49f83e	10000000	4e49f83e	a25e7ed5	ec1786eb
31	ec1786eb					83b1cf9a	6fa64971
32	6fa64971					27f93943	485f7032
33	485f7032					6a94f767	22cb8755
34	22cb8755					c0a69407	e26d1352
35	e26d1352					d19da4e1	33f0b7b3
36	33f0b7b3	f0b7b333	8ca96dc3	20000000	aca96dc3	ec1786eb	40beeb28
37	40beeb28					6fa64971	2f18a259
38	2f18a259					485f7032	6747d26b
39	6747d26b					22cb8755	458c553e
40	458c553e					e26d1352	a7e1466c
41	a7e1466c					33f0b7b3	9411f1df
42	9411f1df	11f1df94	82a19e22	40000000	c2a19e22	40beeb28	821f750a
43	821f750a					2f18a259	ad07d753

44	ad07d753					6747d26b	ca400538
45	ca400538					458c553e	8fcc5006
46	8fcc5006					a7e1466c	282d166a
47	282d166a					9411f1df	bc3ce7b5
48	bc3ce7b5	3ce7b5bc	eb94d565	80000000	6b94d565	821f750a	e98ba06f
49	e98ba06f					ad07d753	448c773c
50	448c773c					ca400538	8ecc7204
51	8ecc7204					8fcc5006	01002202

### А.3 Расширение 256-битного ключа

Здесь показан процесс получения подключей из следующего ключа шифрования:

K = 60 3d eb 10 15 ca 71 be 2b 73 ae f0 85 7d 77 81  
 1f 35 2c 07 3b 61 08 d7 2d 98 10 a3 09 14 df f4

$N_K = 8$ , следовательно

$w_0 = 603deb10, w_1 = 15ca71be, w_2 = 2b73aef0, w_3 = 857d7781$   
 $w_4 = 1f352c07, w_5 = 3b6108d7, w_6 = 2d9810a3, w_7 = 0914dff4$

i (дес)	temp	после RotWord()	после SubWord()	Rcon[i/Nk]	после XOR с Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
8	0914dff4	14dff409	fa9ebf01	01000000	fb9ebf01	603deb10	9ba35411
9	9ba35411					15ca71be	8e6925af
10	8e6925af					2b73aef0	a51a8b5f
11	a51a8b5f					857d7781	2067fcde
12	2067fcde		b785b01d			1f352c07	a8b09c1a
13	a8b09c1a					3b6108d7	93d194cd
14	93d194cd					2d9810a3	be49846e
15	be49846e					0914dff4	b75d5b9a
16	b75d5b9a	5d5b9ab7	4c39b8a9	02000000	4e39b8a9	9ba35411	d59aecb8
17	d59aecb8					8e6925af	5bf3c917
18	5bf3c917					a51a8b5f	fee94248
19	fee94248					2067fcde	de8ebe96
20	de8ebe96		1d19ae90			a8b09c1a	b5a9328a
21	b5a9328a					93d194cd	2678a647
22	2678a647					be49846e	98312229

23	98312229					b75d5b9a	2f6c79b3
24	2f6c79b3	6c79b32f	50b66d15	04000000	54b66d15	d59aecb8	812c81ad
25	812c81ad					5bf3c917	dadf48ba
26	dadf48ba					fee94248	24360af2
27	24360af2					de8ebe96	fab8b464
28	fab8b464		2d6c8d43			b5a9328a	98c5bfc9
29	98c5bfc9					2678a647	bebd198e
30	bebd198e					98312229	268c3ba7
31	268c3ba7					2f6c79b3	09e04214
32	09e04214	e0421409	e12cfa01	08000000	e92cfa01	812c81ad	68007bac
33	68007bac					dadf48ba	b2df3316
34	b2df3316					24360af2	96e939e4
35	96e939e4					fab8b464	6c518d80
36	6c518d80		50d15dcd			98c5bfc9	c814e204
37	c814e204					bebd198e	76a9fb8a
38	76a9fb8a					268c3ba7	5025c02d
39	5025c02d					09e04214	59c58239
40	59c58239	c5823959	a61312cb	10000000	b61312cb	68007bac	de136967
41	de136967					b2df3316	6ccc5a71
42	6ccc5a71					96e939e4	fa256395
43	fa256395					6c518d80	9674ee15
44	9674ee15		90922859			c814e204	5886ca5d
45	5886ca5d					76a9fb8a	2e2f31d7
46	2e2f31d7					5025c02d	7e0af1fa
47	7e0af1fa					59c58239	27cf73c3
48	27cf73c3	cf73c327	8a8f2ecc	20000000	aa8f2ecc	de136967	749c47ab
49	749c47ab					6ccc5a71	18501dda
50	18501dda					fa256395	e2757e4f
51	e2757e4f					9674ee15	7401905a
52	7401905a		927c60be			5886ca5d	cafaaae3
53	cafaaae3					2e2f31d7	e4d59b34
54	e4d59b34					7e0af1fa	9adf6ace
55	9adf6ace					27cf73c3	bd10190d
56	bd10190d	10190dbd	cad4d77a	40000000	8ad4d77a	749c47ab	fe4890d1
57	fe4890d1					18501dda	e6188d0b
58	e6188d0b					e2757e4f	046df344
59	046df344					7401905a	706c631e

## Приложение Б – пример шифрования.

Здесь представлены значения матрицы состояния при шифровании. Длина блока 16 байт и длина ключа 16 байт (то есть  $N_b = 4$  и  $N_K = 4$ ).

Input = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

K = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

Значения ключей раундов взяты из приложения А.

номер раунда	начало раунда	после SubBytes	после ShiftRows	после MixColumns	значение ключа раунда																																																																																
input	<table border="1"> <tr><td>32</td><td>88</td><td>31</td><td>e0</td></tr> <tr><td>43</td><td>5a</td><td>31</td><td>37</td></tr> <tr><td>f6</td><td>30</td><td>98</td><td>07</td></tr> <tr><td>a8</td><td>8d</td><td>a2</td><td>34</td></tr> </table>	32	88	31	e0	43	5a	31	37	f6	30	98	07	a8	8d	a2	34	<table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table>																	<table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table>																	<table border="1"> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td></tr> </table>																	<table border="1"> <tr><td>2b</td><td>28</td><td>ab</td><td>09</td></tr> <tr><td>7e</td><td>ae</td><td>f7</td><td>cf</td></tr> <tr><td>15</td><td>d2</td><td>15</td><td>4f</td></tr> <tr><td>16</td><td>a6</td><td>88</td><td>3c</td></tr> </table>	2b	28	ab	09	7e	ae	f7	cf	15	d2	15	4f	16	a6	88	3c
32	88	31	e0																																																																																		
43	5a	31	37																																																																																		
f6	30	98	07																																																																																		
a8	8d	a2	34																																																																																		
2b	28	ab	09																																																																																		
7e	ae	f7	cf																																																																																		
15	d2	15	4f																																																																																		
16	a6	88	3c																																																																																		
1	<table border="1"> <tr><td>19</td><td>a0</td><td>9a</td><td>e9</td></tr> <tr><td>3d</td><td>f4</td><td>c6</td><td>f8</td></tr> <tr><td>e3</td><td>e2</td><td>8d</td><td>48</td></tr> <tr><td>be</td><td>2b</td><td>2a</td><td>08</td></tr> </table>	19	a0	9a	e9	3d	f4	c6	f8	e3	e2	8d	48	be	2b	2a	08	<table border="1"> <tr><td>d4</td><td>e0</td><td>b8</td><td>1e</td></tr> <tr><td>27</td><td>bf</td><td>b4</td><td>41</td></tr> <tr><td>11</td><td>98</td><td>5d</td><td>52</td></tr> <tr><td>ae</td><td>f1</td><td>e5</td><td>30</td></tr> </table>	d4	e0	b8	1e	27	bf	b4	41	11	98	5d	52	ae	f1	e5	30	<table border="1"> <tr><td>d4</td><td>e0</td><td>b8</td><td>1e</td></tr> <tr><td>bf</td><td>b4</td><td>41</td><td>27</td></tr> <tr><td>5d</td><td>52</td><td>11</td><td>98</td></tr> <tr><td>30</td><td>ae</td><td>f1</td><td>e5</td></tr> </table>	d4	e0	b8	1e	bf	b4	41	27	5d	52	11	98	30	ae	f1	e5	<table border="1"> <tr><td>04</td><td>e0</td><td>48</td><td>28</td></tr> <tr><td>66</td><td>cb</td><td>f8</td><td>06</td></tr> <tr><td>81</td><td>19</td><td>d3</td><td>26</td></tr> <tr><td>e5</td><td>9a</td><td>7a</td><td>4c</td></tr> </table>	04	e0	48	28	66	cb	f8	06	81	19	d3	26	e5	9a	7a	4c	<table border="1"> <tr><td>a0</td><td>88</td><td>23</td><td>2a</td></tr> <tr><td>fa</td><td>54</td><td>a3</td><td>6c</td></tr> <tr><td>fe</td><td>2c</td><td>39</td><td>76</td></tr> <tr><td>17</td><td>b1</td><td>39</td><td>05</td></tr> </table>	a0	88	23	2a	fa	54	a3	6c	fe	2c	39	76	17	b1	39	05
19	a0	9a	e9																																																																																		
3d	f4	c6	f8																																																																																		
e3	e2	8d	48																																																																																		
be	2b	2a	08																																																																																		
d4	e0	b8	1e																																																																																		
27	bf	b4	41																																																																																		
11	98	5d	52																																																																																		
ae	f1	e5	30																																																																																		
d4	e0	b8	1e																																																																																		
bf	b4	41	27																																																																																		
5d	52	11	98																																																																																		
30	ae	f1	e5																																																																																		
04	e0	48	28																																																																																		
66	cb	f8	06																																																																																		
81	19	d3	26																																																																																		
e5	9a	7a	4c																																																																																		
a0	88	23	2a																																																																																		
fa	54	a3	6c																																																																																		
fe	2c	39	76																																																																																		
17	b1	39	05																																																																																		
2	<table border="1"> <tr><td>a4</td><td>68</td><td>6b</td><td>02</td></tr> <tr><td>9c</td><td>9f</td><td>5b</td><td>6a</td></tr> <tr><td>7f</td><td>35</td><td>ea</td><td>50</td></tr> <tr><td>f2</td><td>2b</td><td>43</td><td>49</td></tr> </table>	a4	68	6b	02	9c	9f	5b	6a	7f	35	ea	50	f2	2b	43	49	<table border="1"> <tr><td>49</td><td>45</td><td>7f</td><td>77</td></tr> <tr><td>de</td><td>db</td><td>39</td><td>02</td></tr> <tr><td>d2</td><td>96</td><td>87</td><td>53</td></tr> <tr><td>89</td><td>f1</td><td>1a</td><td>3b</td></tr> </table>	49	45	7f	77	de	db	39	02	d2	96	87	53	89	f1	1a	3b	<table border="1"> <tr><td>49</td><td>45</td><td>7f</td><td>77</td></tr> <tr><td>db</td><td>39</td><td>02</td><td>de</td></tr> <tr><td>87</td><td>53</td><td>d2</td><td>96</td></tr> <tr><td>3b</td><td>89</td><td>f1</td><td>1a</td></tr> </table>	49	45	7f	77	db	39	02	de	87	53	d2	96	3b	89	f1	1a	<table border="1"> <tr><td>58</td><td>1b</td><td>db</td><td>1b</td></tr> <tr><td>4d</td><td>4b</td><td>e7</td><td>6b</td></tr> <tr><td>ca</td><td>5a</td><td>ca</td><td>b0</td></tr> <tr><td>f1</td><td>ac</td><td>a8</td><td>e5</td></tr> </table>	58	1b	db	1b	4d	4b	e7	6b	ca	5a	ca	b0	f1	ac	a8	e5	<table border="1"> <tr><td>f2</td><td>7a</td><td>59</td><td>73</td></tr> <tr><td>c2</td><td>96</td><td>35</td><td>59</td></tr> <tr><td>95</td><td>b9</td><td>80</td><td>f6</td></tr> <tr><td>f2</td><td>43</td><td>7a</td><td>7f</td></tr> </table>	f2	7a	59	73	c2	96	35	59	95	b9	80	f6	f2	43	7a	7f
a4	68	6b	02																																																																																		
9c	9f	5b	6a																																																																																		
7f	35	ea	50																																																																																		
f2	2b	43	49																																																																																		
49	45	7f	77																																																																																		
de	db	39	02																																																																																		
d2	96	87	53																																																																																		
89	f1	1a	3b																																																																																		
49	45	7f	77																																																																																		
db	39	02	de																																																																																		
87	53	d2	96																																																																																		
3b	89	f1	1a																																																																																		
58	1b	db	1b																																																																																		
4d	4b	e7	6b																																																																																		
ca	5a	ca	b0																																																																																		
f1	ac	a8	e5																																																																																		
f2	7a	59	73																																																																																		
c2	96	35	59																																																																																		
95	b9	80	f6																																																																																		
f2	43	7a	7f																																																																																		
3	<table border="1"> <tr><td>aa</td><td>61</td><td>82</td><td>68</td></tr> <tr><td>8f</td><td>dd</td><td>d2</td><td>32</td></tr> <tr><td>5f</td><td>e3</td><td>4a</td><td>46</td></tr> <tr><td>03</td><td>ef</td><td>d2</td><td>9a</td></tr> </table>	aa	61	82	68	8f	dd	d2	32	5f	e3	4a	46	03	ef	d2	9a	<table border="1"> <tr><td>ac</td><td>ef</td><td>13</td><td>45</td></tr> <tr><td>73</td><td>c1</td><td>b5</td><td>23</td></tr> <tr><td>cf</td><td>11</td><td>d6</td><td>5a</td></tr> <tr><td>7b</td><td>df</td><td>b5</td><td>b8</td></tr> </table>	ac	ef	13	45	73	c1	b5	23	cf	11	d6	5a	7b	df	b5	b8	<table border="1"> <tr><td>ac</td><td>ef</td><td>13</td><td>45</td></tr> <tr><td>c1</td><td>b5</td><td>23</td><td>73</td></tr> <tr><td>d6</td><td>5a</td><td>cf</td><td>11</td></tr> <tr><td>b8</td><td>7b</td><td>df</td><td>b5</td></tr> </table>	ac	ef	13	45	c1	b5	23	73	d6	5a	cf	11	b8	7b	df	b5	<table border="1"> <tr><td>75</td><td>20</td><td>53</td><td>bb</td></tr> <tr><td>ec</td><td>0b</td><td>c0</td><td>25</td></tr> <tr><td>09</td><td>63</td><td>cf</td><td>d0</td></tr> <tr><td>93</td><td>33</td><td>7c</td><td>dc</td></tr> </table>	75	20	53	bb	ec	0b	c0	25	09	63	cf	d0	93	33	7c	dc	<table border="1"> <tr><td>3d</td><td>47</td><td>1e</td><td>6d</td></tr> <tr><td>80</td><td>16</td><td>23</td><td>7a</td></tr> <tr><td>47</td><td>fe</td><td>7e</td><td>88</td></tr> <tr><td>7d</td><td>3e</td><td>44</td><td>3b</td></tr> </table>	3d	47	1e	6d	80	16	23	7a	47	fe	7e	88	7d	3e	44	3b
aa	61	82	68																																																																																		
8f	dd	d2	32																																																																																		
5f	e3	4a	46																																																																																		
03	ef	d2	9a																																																																																		
ac	ef	13	45																																																																																		
73	c1	b5	23																																																																																		
cf	11	d6	5a																																																																																		
7b	df	b5	b8																																																																																		
ac	ef	13	45																																																																																		
c1	b5	23	73																																																																																		
d6	5a	cf	11																																																																																		
b8	7b	df	b5																																																																																		
75	20	53	bb																																																																																		
ec	0b	c0	25																																																																																		
09	63	cf	d0																																																																																		
93	33	7c	dc																																																																																		
3d	47	1e	6d																																																																																		
80	16	23	7a																																																																																		
47	fe	7e	88																																																																																		
7d	3e	44	3b																																																																																		
4	<table border="1"> <tr><td>48</td><td>67</td><td>4d</td><td>d6</td></tr> <tr><td>6c</td><td>1d</td><td>e3</td><td>5f</td></tr> <tr><td>4e</td><td>9d</td><td>b1</td><td>58</td></tr> <tr><td>ee</td><td>0d</td><td>38</td><td>e7</td></tr> </table>	48	67	4d	d6	6c	1d	e3	5f	4e	9d	b1	58	ee	0d	38	e7	<table border="1"> <tr><td>52</td><td>85</td><td>e3</td><td>f6</td></tr> <tr><td>50</td><td>a4</td><td>11</td><td>cf</td></tr> <tr><td>2f</td><td>5e</td><td>c8</td><td>6a</td></tr> <tr><td>28</td><td>d7</td><td>07</td><td>94</td></tr> </table>	52	85	e3	f6	50	a4	11	cf	2f	5e	c8	6a	28	d7	07	94	<table border="1"> <tr><td>52</td><td>85</td><td>e3</td><td>f6</td></tr> <tr><td>a4</td><td>11</td><td>cf</td><td>50</td></tr> <tr><td>c8</td><td>6a</td><td>2f</td><td>5e</td></tr> <tr><td>94</td><td>28</td><td>d7</td><td>07</td></tr> </table>	52	85	e3	f6	a4	11	cf	50	c8	6a	2f	5e	94	28	d7	07	<table border="1"> <tr><td>0f</td><td>60</td><td>6f</td><td>5e</td></tr> <tr><td>d6</td><td>31</td><td>c0</td><td>b3</td></tr> <tr><td>da</td><td>38</td><td>10</td><td>13</td></tr> <tr><td>a9</td><td>bf</td><td>6b</td><td>01</td></tr> </table>	0f	60	6f	5e	d6	31	c0	b3	da	38	10	13	a9	bf	6b	01	<table border="1"> <tr><td>ef</td><td>a8</td><td>b6</td><td>db</td></tr> <tr><td>44</td><td>52</td><td>71</td><td>0b</td></tr> <tr><td>a5</td><td>5b</td><td>25</td><td>ad</td></tr> <tr><td>41</td><td>7f</td><td>3b</td><td>00</td></tr> </table>	ef	a8	b6	db	44	52	71	0b	a5	5b	25	ad	41	7f	3b	00
48	67	4d	d6																																																																																		
6c	1d	e3	5f																																																																																		
4e	9d	b1	58																																																																																		
ee	0d	38	e7																																																																																		
52	85	e3	f6																																																																																		
50	a4	11	cf																																																																																		
2f	5e	c8	6a																																																																																		
28	d7	07	94																																																																																		
52	85	e3	f6																																																																																		
a4	11	cf	50																																																																																		
c8	6a	2f	5e																																																																																		
94	28	d7	07																																																																																		
0f	60	6f	5e																																																																																		
d6	31	c0	b3																																																																																		
da	38	10	13																																																																																		
a9	bf	6b	01																																																																																		
ef	a8	b6	db																																																																																		
44	52	71	0b																																																																																		
a5	5b	25	ad																																																																																		
41	7f	3b	00																																																																																		
5	<table border="1"> <tr><td>e0</td><td>c8</td><td>d9</td><td>85</td></tr> <tr><td>92</td><td>63</td><td>b1</td><td>b8</td></tr> <tr><td>7f</td><td>63</td><td>35</td><td>be</td></tr> <tr><td>e8</td><td>c0</td><td>50</td><td>01</td></tr> </table>	e0	c8	d9	85	92	63	b1	b8	7f	63	35	be	e8	c0	50	01	<table border="1"> <tr><td>e1</td><td>e8</td><td>35</td><td>97</td></tr> <tr><td>4f</td><td>fb</td><td>c8</td><td>6c</td></tr> <tr><td>d2</td><td>fb</td><td>96</td><td>ae</td></tr> <tr><td>9b</td><td>ba</td><td>53</td><td>7c</td></tr> </table>	e1	e8	35	97	4f	fb	c8	6c	d2	fb	96	ae	9b	ba	53	7c	<table border="1"> <tr><td>e1</td><td>e8</td><td>35</td><td>97</td></tr> <tr><td>fb</td><td>c8</td><td>6c</td><td>4f</td></tr> <tr><td>96</td><td>ae</td><td>d2</td><td>fb</td></tr> <tr><td>7c</td><td>9b</td><td>ba</td><td>53</td></tr> </table>	e1	e8	35	97	fb	c8	6c	4f	96	ae	d2	fb	7c	9b	ba	53	<table border="1"> <tr><td>25</td><td>bd</td><td>b6</td><td>4c</td></tr> <tr><td>d1</td><td>11</td><td>3a</td><td>4c</td></tr> <tr><td>a9</td><td>d1</td><td>33</td><td>c0</td></tr> <tr><td>ad</td><td>68</td><td>8e</td><td>b0</td></tr> </table>	25	bd	b6	4c	d1	11	3a	4c	a9	d1	33	c0	ad	68	8e	b0	<table border="1"> <tr><td>d4</td><td>7c</td><td>ca</td><td>11</td></tr> <tr><td>d1</td><td>83</td><td>f2</td><td>f9</td></tr> <tr><td>c6</td><td>9d</td><td>b8</td><td>15</td></tr> <tr><td>f8</td><td>87</td><td>bc</td><td>bc</td></tr> </table>	d4	7c	ca	11	d1	83	f2	f9	c6	9d	b8	15	f8	87	bc	bc
e0	c8	d9	85																																																																																		
92	63	b1	b8																																																																																		
7f	63	35	be																																																																																		
e8	c0	50	01																																																																																		
e1	e8	35	97																																																																																		
4f	fb	c8	6c																																																																																		
d2	fb	96	ae																																																																																		
9b	ba	53	7c																																																																																		
e1	e8	35	97																																																																																		
fb	c8	6c	4f																																																																																		
96	ae	d2	fb																																																																																		
7c	9b	ba	53																																																																																		
25	bd	b6	4c																																																																																		
d1	11	3a	4c																																																																																		
a9	d1	33	c0																																																																																		
ad	68	8e	b0																																																																																		
d4	7c	ca	11																																																																																		
d1	83	f2	f9																																																																																		
c6	9d	b8	15																																																																																		
f8	87	bc	bc																																																																																		

6

f1	c1	7c	5d
00	92	c8	b5
6f	4c	8b	d5
55	ef	32	0c

a1	78	10	4c
63	4f	e8	d5
a8	29	3d	03
fc	df	23	fe

a1	78	10	4c
4f	e8	d5	63
3d	03	a8	29
fe	fc	df	23

4b	2c	33	37
86	4a	9d	d2
8d	89	f4	18
6d	80	e8	d8

 $\oplus$ 

6d	11	db	ca
88	0b	f9	00
a3	3e	86	93
7a	fd	41	fd

=

7

26	3d	e8	fd
0e	41	64	d2
2e	b7	72	8b
17	7d	a9	25

f7	27	9b	54
ab	83	43	b5
31	a9	40	3d
f0	ff	d3	3f

f7	27	9b	54
83	43	b5	ab
40	3d	31	a9
3f	f0	ff	d3

14	46	27	34
15	16	46	2a
b5	15	56	d8
bf	ec	d7	43

 $\oplus$ 

4e	5f	84	4e
54	5f	a6	a6
f7	c9	4f	dc
0e	f3	b2	4f

=

8

5a	19	a3	7a
41	49	e0	8c
42	dc	19	04
b1	1f	65	0c

be	d4	0a	da
83	3b	e1	64
2c	86	d4	f2
c8	c0	4d	fe

be	d4	0a	da
3b	e1	64	83
d4	f2	2c	86
fe	c8	c0	4d

00	b1	54	fa
51	c8	76	1b
2f	89	6d	99
d1	ff	cd	ea

 $\oplus$ 

ea	b5	31	7f
d2	8d	2b	8d
73	ba	f5	29
21	d2	60	2f

=

9

ea	04	65	85
83	45	5d	96
5c	33	98	b0
f0	2d	ad	c5

87	f2	4d	97
ec	6e	4c	90
4a	c3	46	e7
8c	d8	95	a6

87	f2	4d	97
6e	4c	90	ec
46	e7	4a	c3
a6	8c	d8	95

47	40	a3	4c
37	d4	70	9f
94	e4	3a	42
ed	a5	a6	bc

 $\oplus$ 

ac	19	28	57
77	fa	d1	5c
66	dc	29	00
f3	21	41	6e

=

10

eb	59	8b	1b
40	2e	a1	c3
f2	38	13	42
1e	84	e7	d2

e9	cb	3d	af
09	31	32	2e
89	07	7d	2c
72	5f	94	b5

e9	cb	3d	af
31	32	2e	09
7d	2c	89	07
b5	72	5f	94


 $\oplus$ 

d0	c9	e1	b6
14	ee	3f	63
f9	25	0c	0c
a8	89	c8	a6

=

output

39	02	dc	19
25	dc	11	6a
84	09	85	0b
1d	fb	97	32

## Приложение В – примеры промежуточных вычислений.

Это приложение содержит результаты промежуточных вычислений для всех трёх длин ключа шифрования ( $N_K = 4, 6$  или  $8$ ) при выполнении шифрования, расшифрования и расшифрования с помощью эквивалентной процедуры расшифрования описанных в подразделах 5.1, 5.3 и пункте 5.3.5 соответственно. Дополнительные примеры можно найти, обратившись по ссылкам [1] и [5].

Промежуточные результаты представлены в виде одномерных массивов – векторов. Значения всех векторов записаны в 16-ричном виде, где каждая пара символов представляет значение байта. При этом левый символ каждой пары обозначает 4-х битовую комбинацию старших битов байта (см. подраздел 3.2). Правый символ обозначает 4-х битовую комбинацию младших битов байта. В данных векторах номер байта (группы из двух 16-ричных чисел) в пределах вектора начинается от нуля и увеличивается слева направо.

Условные обозначения при шифровании (номер раунда от 0 до 10, 12 или 14)

input: вход процедуры шифрования

start: значение матрицы состояния в начале раунда [r]

s\_box: значение матрицы состояния после выполнения процедуры SubBytes

s\_row: значение матрицы состояния после выполнения процедуры ShiftRows

m\_col: значение матрицы состояния после выполнения процедуры MixColumns

k\_sch: значение подключа для раунда [r]

output: результат шифрования

Условные обозначения при расшифровании (номер раунда от 0 до 10, 12 или 14)

iinput: вход процедуры расшифрования

istart: значение матрицы состояния в начале раунда [r]

is\_box: значение матрицы состояния после выполнения процедуры InvSubBytes

is\_row: значение матрицы состояния после выполнения процедуры InvShiftRows

ik\_sch: значение подключа для раунда [r]

ik\_add: значение матрицы состояния после выполнения процедуры AddRoundKey

ioutput: результат расшифрования

Условные обозначения при расшифровании с помощью эквивалентной процедуры расшифрования (номер раунда от 0 до 10, 12 или 14)

iinput: вход процедуры расшифрования

istart: значение матрицы состояния в начале раунда [r]

is\_box: значение матрицы состояния после выполнения процедуры InvSubBytes

is\_row: значение матрицы состояния после выполнения процедуры InvShiftRows

im\_col: значение матрицы состояния после выполнения процедуры InvMixColumns

ik\_sch: значение подключа для раунда [r]

ioutput: результат расшифрования

## B.1 AES-128 ( $Nk=4, Nr=10$ )

открытый текст: 00112233445566778899aabbccddeeff  
ключ: 000102030405060708090a0b0c0d0e0f

### ШИФРОВАНИЕ:

```
round[ 0].input      00112233445566778899aabbccddeeff
round[ 0].k_sch      000102030405060708090a0b0c0d0e0f
round[ 1].start      00102030405060708090a0b0c0d0e0f0
round[ 1].s_box      63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row      6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col      5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch      d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 2].start      89d810e8855ace682d1843d8cb128fe4
round[ 2].s_box      a761ca9b97be8b45d8ad1a611fc97369
round[ 2].s_row      a7be1a6997ad739bd8c9ca451f618b61
round[ 2].m_col      ff87968431d86a51645151fa773ad009
round[ 2].k_sch      b692cf0b643dbdf1be9bc5006830b3fe
round[ 3].start      4915598f55e5d7a0daca94fa1f0a63f7
round[ 3].s_box      3b59cb73fcd90ee05774222dc067fb68
round[ 3].s_row      3bd92268fc74fb735767cbe0c0590e2d
round[ 3].m_col      4c9c1e66f771f0762c3f868e534df256
round[ 3].k_sch      b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 4].start      fa636a2825b339c940668a3157244d17
round[ 4].s_box      2dfb02343f6d12dd09337ec75b36e3f0
round[ 4].s_row      2d6d7ef03f33e334093602dd5bfb12c7
round[ 4].m_col      6385b79ffc538df997be478e7547d691
round[ 4].k_sch      47f7f7bc95353e03f96c32bcfd058dfd
round[ 5].start      247240236966b3fa6ed2753288425b6c
round[ 5].s_box      36400926f9336d2d9fb59d23c42c3950
round[ 5].s_row      36339d50f9b539269f2c092dc4406d23
round[ 5].m_col      f4bcd45432e554d075f1d6c51dd03b3c
round[ 5].k_sch      3caaa3e8a99f9deb50f3af57adf622aa
round[ 6].start      c81677bc9b7ac93b25027992b0261996
round[ 6].s_box      e847f56514dadde23f77b64fe7f7d490
round[ 6].s_row      e8dab6901477d4653ff7f5e2e747dd4f
round[ 6].m_col      9816ee7400f87f556b2c049c8e5ad036
round[ 6].k_sch      5e390f7df7a69296a7553dc10aa31f6b
round[ 7].start      c62fe109f75eedc3cc79395d84f9cf5d
round[ 7].s_box      b415f8016858552e4bb6124c5f998a4c
round[ 7].s_row      b458124c68b68a014b99f82e5f15554c
round[ 7].m_col      c57e1c159a9bd286f05f4be098c63439
round[ 7].k_sch      14f9701ae35fe28c440adf4d4ea9c026
round[ 8].start      d1876c0f79c4300ab45594add66ff41f
round[ 8].s_box      3e175076b61c04678dfc2295f6a8bfc0
round[ 8].s_row      3e1c22c0b6fcbf768da85067f6170495
round[ 8].m_col      baa03de7a1f9b56ed5512cba5f414d23
round[ 8].k_sch      47438735a41c65b9e016baf4aebf7ad2
round[ 9].start      fde3bad205e5d0d73547964ef1fe37f1
round[ 9].s_box      5411f4b56bd9700e96a0902fa1bb9aa1
round[ 9].s_row      54d990a16ba09ab596bbf40ea111702f
round[ 9].m_col      e9f74eec023020f61bf2ccf2353c21c7
round[ 9].k_sch      549932d1f08557681093ed9cbe2c974e
round[10].start      bd6e7c3df2b5779e0b61216e8b10b689
round[10].s_box      7a9f102789d5f50b2beffd9f3dca4ea7
round[10].s_row      7ad5fda789ef4e272bca100b3d9ff59f
round[10].k_sch      13111d7fe3944a17f307a78b4d2b30c5
round[10].output     69c4e0d86a7b0430d8cdb78070b4c55a
```

## РАСШИФРОВАНИЕ:

```
round[ 0].iinput      69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0].ik_sch     13111d7fe3944a17f307a78b4d2b30c5
round[ 1].istart     7ad5fda789ef4e272bca100b3d9ff59f
round[ 1].is_row    7a9f102789d5f50b2beffd9f3dca4ea7
round[ 1].is_box    bd6e7c3df2b5779e0b61216e8b10b689
round[ 1].ik_sch    549932d1f08557681093ed9cbe2c974e
round[ 1].ik_add    e9f74eec023020f61bf2ccf2353c21c7
round[ 2].istart     54d990a16ba09ab596bbf40ea111702f
round[ 2].is_row    5411f4b56bd9700e96a0902fa1bb9aa1
round[ 2].is_box    fde3bad205e5d0d73547964ef1fe37f1
round[ 2].ik_sch    47438735a41c65b9e016baf4aebf7ad2
round[ 2].ik_add    baa03de7a1f9b56ed5512cba5f414d23
round[ 3].istart     3e1c22c0b6fcfbf768da85067f6170495
round[ 3].is_row    3e175076b61c04678dfc2295f6a8bfc0
round[ 3].is_box    d1876c0f79c4300ab45594add66ff41f
round[ 3].ik_sch    14f9701ae35fe28c440adf4d4ea9c026
round[ 3].ik_add    c57e1c159a9bd286f05f4be098c63439
round[ 4].istart     b458124c68b68a014b99f82e5f15554c
round[ 4].is_row    b415f8016858552e4bb6124c5f998a4c
round[ 4].is_box    c62fe109f75eedc3cc79395d84f9cf5d
round[ 4].ik_sch    5e390f7df7a69296a7553dc10aa31f6b
round[ 4].ik_add    9816ee7400f87f556b2c049c8e5ad036
round[ 5].istart     e8dab6901477d4653ff7f5e2e747dd4f
round[ 5].is_row    e847f56514dadde23f77b64fe7f7d490
round[ 5].is_box    c81677bc9b7ac93b25027992b0261996
round[ 5].ik_sch    3caaa3e8a99f9deb50f3af57adf622aa
round[ 5].ik_add    f4bcd45432e554d075f1d6c51dd03b3c
round[ 6].istart     36339d50f9b539269f2c092dc4406d23
round[ 6].is_row    36400926f9336d2d9fb59d23c42c3950
round[ 6].is_box    247240236966b3fa6ed2753288425b6c
round[ 6].ik_sch    47f7f7bc95353e03f96c32bcfd058dfd
round[ 6].ik_add    6385b79ffc538df997be478e7547d691
round[ 7].istart     2d6d7ef03f33e334093602dd5bfb12c7
round[ 7].is_row    2dfb02343f6d12dd09337ec75b36e3f0
round[ 7].is_box    fa636a2825b339c940668a3157244d17
round[ 7].ik_sch    b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 7].ik_add    4c9c1e66f771f0762c3f868e534df256
round[ 8].istart     3bd92268fc74fb735767cbe0c0590e2d
round[ 8].is_row    3b59cb73fcd90ee05774222dc067fb68
round[ 8].is_box    4915598f55e5d7a0daca94fa1f0a63f7
round[ 8].ik_sch    b692cf0b643dbdf1be9bc5006830b3fe
round[ 8].ik_add    ff87968431d86a51645151fa773ad009
round[ 9].istart     a7be1a6997ad739bd8c9ca451f618b61
round[ 9].is_row    a761ca9b97be8b45d8ad1a611fc97369
round[ 9].is_box    89d810e8855ace682d1843d8cb128fe4
round[ 9].ik_sch    d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 9].ik_add    5f72641557f5bc92f7be3b291db9f91a
round[10].istart     6353e08c0960e104cd70b751bacad0e7
round[10].is_row    63cab7040953d051cd60e0e7ba70e18c
round[10].is_box    00102030405060708090a0b0c0d0e0f0
round[10].ik_sch    000102030405060708090a0b0c0d0e0f
round[10].ioutput   00112233445566778899aabbccddeeff
```

ЭКВИВАЛЕНТНАЯ ПРОЦЕДУРА  
РАСШИФРОВАНИЯ:

```
round[ 0].iinput      69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0].ik_sch     13111d7fe3944a17f307a78b4d2b30c5
round[ 1].istart     7ad5fda789ef4e272bca100b3d9ff59f
round[ 1].is_box     bdb52189f261b63d0b107c9e8b6e776e
round[ 1].is_row     bd6e7c3df2b5779e0b61216e8b10b689
round[ 1].im_col     4773b91ff72f354361cb018ea1e6cf2c
round[ 1].ik_sch     13aa29be9c8faff6f770f58000f7bf03
round[ 2].istart     54d990a16ba09ab596bbf40ea111702f
round[ 2].is_box     fde596f1054737d235febad7f1e3d04e
round[ 2].is_row     fde3bad205e5d0d73547964ef1fe37f1
round[ 2].im_col     2d7e86a339d9393ee6570a1101904e16
round[ 2].ik_sch     1362a4638f2586486bff5a76f7874a83
round[ 3].istart     3e1c22c0b6fcbf768da85067f6170495
round[ 3].is_box     d1c4941f7955f40fb46f6c0ad68730ad
round[ 3].is_row     d1876c0f79c4300ab45594add66ff41f
round[ 3].im_col     39daee38f4f1a82aaf432410c36d45b9
round[ 3].ik_sch     8d82fc749c47222be4dad3e9c7810f5
round[ 4].istart     b458124c68b68a014b99f82e5f15554c
round[ 4].is_box     c65e395df779cf09ccf9e1c3842fed5d
round[ 4].is_row     c62fe109f75eedc3cc79395d84f9cf5d
round[ 4].im_col     9a39b1d05b20a3a476a0bf79fe51184
round[ 4].ik_sch     72e3098d11c5de5f789dfe1578a2cccb
round[ 5].istart     e8dab6901477d4653ff7f5e2e747dd4f
round[ 5].is_box     c87a79969b0219bc2526773bb016c992
round[ 5].is_row     c81677bc9b7ac93b25027992b0261996
round[ 5].im_col     18f78d779a93eef4f6742967c47f5ffd
round[ 5].ik_sch     2ec410276326d7d26958204a003f32de
round[ 6].istart     36339d50f9b539269f2c092dc4406d23
round[ 6].is_box     2466756c69d25b236e4240fa8872b332
round[ 6].is_row     247240236966b3fa6ed2753288425b6c
round[ 6].im_col     85cf8bf472d124c10348f545329c0053
round[ 6].ik_sch     a8a2f5044de2c7f50a7ef79869671294
round[ 7].istart     2d6d7ef03f33e334093602dd5bfb12c7
round[ 7].is_box     fab38a1725664d2840246ac957633931
round[ 7].is_row     fa636a2825b339c940668a3157244d17
round[ 7].im_col     fc1fc1f91934c98210fbfb8da340eb21
round[ 7].ik_sch     c7c6e391e54032f1479c306d6319e50c
round[ 8].istart     3bd92268fc74fb735767cbe0c0590e2d
round[ 8].is_box     49e594f755ca638fda0a59a01f15d7fa
round[ 8].is_row     4915598f55e5d7a0daca94fa1f0a63f7
round[ 8].im_col     076518f0b52ba2fb7a15c8d93be45e00
round[ 8].ik_sch     a0db02992286d160a2dc029c2485d561
round[ 9].istart     a7be1a6997ad739bd8c9ca451f618b61
round[ 9].is_box     895a43e485188fe82d121068cbd8ced8
round[ 9].is_row     89d810e8855ace682d1843d8cb128fe4
round[ 9].im_col     ef053f7c8b3d32fd4d2a64ad3c93071a
round[ 9].ik_sch     8c56dff0825dd3f9805ad3fc8659d7fd
round[10].istart     6353e08c0960e104cd70b751bacad0e7
round[10].is_box     0050a0f04090e03080d02070c01060b0
round[10].is_row     00102030405060708090a0b0c0d0e0f0
round[10].ik_sch     000102030405060708090a0b0c0d0e0f
round[10].ioutput    00112233445566778899aabbccddeeff
```

## B.2 AES-192 ( $Nk=6, Nr=12$ )

открытый текст: 00112233445566778899aabbccddeeff

ключ: 000102030405060708090a0b0c0d0e0f1011121314151617

ШИФРОВАНИЕ:

```
round[ 0].input      00112233445566778899aabbccddeeff
round[ 0].k_sch      000102030405060708090a0b0c0d0e0f
round[ 1].start      00102030405060708090a0b0c0d0e0f0
round[ 1].s_box      63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row      6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col      5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch      10111213141516175846f2f95c43f4fe
round[ 2].start      4f63760643e0aa85aff8c9d041fa0de4
round[ 2].s_box      84fb386f1ae1ac977941dd70832dd769
round[ 2].s_row      84e1dd691a41d76f792d389783fbac70
round[ 2].m_col      9f487f794f955f662afc86abd7f1ab29
round[ 2].k_sch      544afef55847f0fa4856e2e95c43f4fe
round[ 3].start      cb02818c17d2af9c62aa64428bb25fd7
round[ 3].s_box      1f770c64f0b579deaaac432c3d37cf0e
round[ 3].s_row      1fb5430ef0accf64aa370cde3d77792c
round[ 3].m_col      b7a53ecbbf9d75a0c40efc79b674cc11
round[ 3].k_sch      40f949b31cbabd4d48f043b810b7b342
round[ 4].start      f75c7778a327c8ed8cfefbc1a6c37f53
round[ 4].s_box      684af5bc0acce85564bb0878242ed2ed
round[ 4].s_row      68cc08ed0abbd2bc642ef555244ae878
round[ 4].m_col      7a1e98bdacb6d1141a6944dd06eb2d3e
round[ 4].k_sch      58e151ab04a2a5557effb5416245080c
round[ 5].start      22ffc916a81474416496f19c64ae2532
round[ 5].s_box      9316dd47c2fa92834390a1de43e43f23
round[ 5].s_row      93faa123c2903f4743e4dd83431692de
round[ 5].m_col      aaa755b34cffe57cef6f98e1f01c13e6
round[ 5].k_sch      2ab54bb43a02f8f662e3a95d66410c08
round[ 6].start      80121e0776fd1d8a8d8c31bc965d1fee
round[ 6].s_box      cdc972c53854a47e5d64c765904cc028
round[ 6].s_row      cd54c7283864c0c55d4c727e90c9a465
round[ 6].m_col      921f748fd96e937d622d7725ba8ba50c
round[ 6].k_sch      f501857297448d7ebdf1c6ca87f33e3c
round[ 7].start      671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 7].s_box      8572a1542fe5727b9e86c8df27bc1404
round[ 7].s_row      85e5c8042f8614549ebca17b277272df
round[ 7].m_col      e913e7b18f507d4b227ef652758acbcc
round[ 7].k_sch      e510976183519b6934157c9ea351f1e0
round[ 8].start      0c0370d00c01e622166b8accd6db3a2c
round[ 8].s_box      fe7b5170fe7c8e93477f7e4bf6b98071
round[ 8].s_row      fe7c7e71fe7f807047b95193f67b8e4b
round[ 8].m_col      6cf5edf996eb0a069c4ef21cbfc25762
round[ 8].k_sch      1ea0372a995309167c439e77ff12051e
round[ 9].start      7255dad30fb80310e00d6c6b40d0527c
round[ 9].s_box      40fc5766766c7bcae1d7507f09700010
round[ 9].s_row      406c501076d70066e17057ca09fc7b7f
round[ 9].m_col      7478bcdce8a50b81d4327a9009188262
round[ 9].k_sch      dd7e0e887e2fff68608fc842f9dcc154
round[10].start     a906b254968af4e9b4bdb2d2f0c44336
round[10].s_box     d36f3720907ebf1e8d7a37b58c1c1a05
round[10].s_row     d37e3705907a1a208d1c371e8c6fbfb5
round[10].m_col     0d73cc2d8f6abe8b0cf2dd9bb83d422e
round[10].k_sch     859f5f237a8d5a3dc0c02952beefd63a
```

```

round[11].start      88ec930ef5e7e4b6cc32f4c906d29414
round[11].s_box      c4cedcabe694694e4b23bfdd6fb522fa
round[11].s_row      c494bffae62322ab4bb5dc4e6fce69dd
round[11].m_col      71d720933b6d677dc00b8f28238e0fb7
round[11].k_sch      de601e7827bcd2ca223800fd8aeda32
round[12].start      afb73eeb1cd1b85162280f27fb20d585
round[12].s_box      79a9b2e99c3e6cd1aa3476cc0fb70397
round[12].s_row      793e76979c3403e9aab7b2d10fa96ccc
round[12].k_sch      a4970a331a78dc09c418c271e3a41d5d
round[12].output     dda97ca4864cdfe06eaf70a0ec0d7191

```

**РАСШИФРОВАНИЕ:**

```

round[ 0].iinput     dda97ca4864cdfe06eaf70a0ec0d7191
round[ 0].ik_sch     a4970a331a78dc09c418c271e3a41d5d
round[ 1].istart     793e76979c3403e9aab7b2d10fa96ccc
round[ 1].is_row     79a9b2e99c3e6cd1aa3476cc0fb70397
round[ 1].is_box     afb73eeb1cd1b85162280f27fb20d585
round[ 1].ik_sch     de601e7827bcd2ca223800fd8aeda32
round[ 1].ik_add     71d720933b6d677dc00b8f28238e0fb7
round[ 2].istart     c494bffae62322ab4bb5dc4e6fce69dd
round[ 2].is_row     c4cedcabe694694e4b23bfdd6fb522fa
round[ 2].is_box     88ec930ef5e7e4b6cc32f4c906d29414
round[ 2].ik_sch     859f5f237a8d5a3dc0c02952beefd63a
round[ 2].ik_add     0d73cc2d8f6abe8b0cf2dd9bb83d422e
round[ 3].istart     d37e3705907a1a208d1c371e8c6fbfb5
round[ 3].is_row     d36f3720907ebf1e8d7a37b58c1c1a05
round[ 3].is_box     a906b254968af4e9b4bdb2d2f0c44336
round[ 3].ik_sch     dd7e0e887e2fff68608fc842f9dcc154
round[ 3].ik_add     7478bcdce8a50b81d4327a9009188262
round[ 4].istart     406c501076d70066e17057ca09fc7b7f
round[ 4].is_row     40fc5766766c7bcae1d7507f09700010
round[ 4].is_box     7255dad30fb80310e00d6c6b40d0527c
round[ 4].ik_sch     1ea0372a995309167c439e77ff12051e
round[ 4].ik_add     6cf5edf996eb0a069c4ef21cbfc25762
round[ 5].istart     fe7c7e71fe7f807047b95193f67b8e4b
round[ 5].is_row     fe7b5170fe7c8e93477f7e4bf6b98071
round[ 5].is_box     0c0370d00c01e622166b8accd6db3a2c
round[ 5].ik_sch     e510976183519b6934157c9ea351f1e0
round[ 5].ik_add     e913e7b18f507d4b227ef652758acbcc
round[ 6].istart     85e5c8042f8614549ebca17b277272df
round[ 6].is_row     8572a1542fe5727b9e86c8df27bc1404
round[ 6].is_box     671ef1fd4e2a1e03dfdcb1ef3d789b30
round[ 6].ik_sch     f501857297448d7ebdf1c6ca87f33e3c
round[ 6].ik_add     921f748fd96e937d622d7725ba8ba50c
round[ 7].istart     cd54c7283864c0c55d4c727e90c9a465
round[ 7].is_row     cdc972c53854a47e5d64c765904cc028
round[ 7].is_box     80121e0776fd1d8a8d8c31bc965d1fee
round[ 7].ik_sch     2ab54bb43a02f8f662e3a95d66410c08
round[ 7].ik_add     aaa755b34cffe57cef6f98e1f01c13e6

```

```

round[ 8].istart 93faa123c2903f4743e4dd83431692de
round[ 8].is_row 9316dd47c2fa92834390a1de43e43f23
round[ 8].is_box 22ffc916a81474416496f19c64ae2532
round[ 8].ik_sch 58e151ab04a2a5557effb5416245080c
round[ 8].ik_add 7a1e98bdacb6d1141a6944dd06eb2d3e
round[ 9].istart 68cc08ed0abbd2bc642ef555244ae878
round[ 9].is_row 684af5bc0acce85564bb0878242ed2ed
round[ 9].is_box f75c7778a327c8ed8cfefbfc1a6c37f53
round[ 9].ik_sch 40f949b31cbabd4d48f043b810b7b342
round[ 9].ik_add b7a53ecbbf9d75a0c40efc79b674cc11
round[10].istart 1fb5430ef0accf64aa370cde3d77792c
round[10].is_row 1f770c64f0b579deaaac432c3d37cf0e
round[10].is_box cb02818c17d2af9c62aa64428bb25fd7
round[10].ik_sch 544afef55847f0fa4856e2e95c43f4fe
round[10].ik_add 9f487f794f955f662afc86abd7f1ab29
round[11].istart 84e1dd691a41d76f792d389783fbac70
round[11].is_row 84fb386f1ae1ac977941dd70832dd769
round[11].is_box 4f63760643e0aa85aff8c9d041fa0de4
round[11].ik_sch 10111213141516175846f2f95c43f4fe
round[11].ik_add 5f72641557f5bc92f7be3b291db9f91a
round[12].istart 6353e08c0960e104cd70b751bacad0e7
round[12].is_row 63cab7040953d051cd60e0e7ba70e18c
round[12].is_box 00102030405060708090a0b0c0d0e0f0
round[12].ik_sch 000102030405060708090a0b0c0d0e0f
round[12].ioutput 00112233445566778899aabbccddeeff

```

**ЭКВИВАЛЕНТНАЯ ПРОЦЕДУРА  
РАСШИФРОВАНИЯ:**

```

round[ 0].iinput dda97ca4864cdfe06eaf70a0ec0d7191
round[ 0].ik_sch a4970a331a78dc09c418c271e3a41d5d
round[ 1].istart 793e76979c3403e9aab7b2d10fa96ccc
round[ 1].is_box afd10f851c28d5eb62203e51fbb7b827
round[ 1].is_row afb73eeb1cd1b85162280f27fb20d585
round[ 1].im_col 122a02f7242ac8e20605afce51cc7264
round[ 1].ik_sch d6bebd0dc209ea494db073803e021bb9
round[ 2].istart c494bffae62322ab4bb5dc4e6fce69dd
round[ 2].is_box 88e7f414f532940eccd293b606ece4c9
round[ 2].is_row 88ec930ef5e7e4b6cc32f4c906d29414
round[ 2].im_col 5cc7aecce3c872194ae5ef8309a933c7
round[ 2].ik_sch 8fb999c973b26839c7f9d89d85c68c72
round[ 3].istart d37e3705907a1a208d1c371e8c6fbfb5
round[ 3].is_box a98ab23696bd4354b4c4b2e9f006f4d2
round[ 3].is_row a906b254968af4e9b4bdb2d2f0c44336
round[ 3].im_col b7113ed134e85489b20866b51d4b2c3b
round[ 3].ik_sch f77d6ec1423f54ef5378317f14b75744
round[ 4].istart 406c501076d70066e17057ca09fc7b7f
round[ 4].is_box 72b86c7c0f0d52d3e0d0da104055036b
round[ 4].is_row 7255dad30fb80310e00d6c6b40d0527c
round[ 4].im_col ef3b1be1b9b0e64bdcb79f1e0a707fbb
round[ 4].ik_sch 1147659047cf663b9b0ece8dfc0bf1f0

```

```

round[ 5].istart      fe7c7e71fe7f807047b95193f67b8e4b
round[ 5].is_box     0c018a2c0c6b3ad016db7022d603e6cc
round[ 5].is_row     0c0370d00c01e622166b8accd6db3a2c
round[ 5].im_col     592460b248832b2952e0b831923048f1
round[ 5].ik_sch     dcc1a8b667053f7dcc5c194ab5423a2e
round[ 6].istart      85e5c8042f8614549ebca17b277272df
round[ 6].is_box     672ab1304edc9bfddf78f1033d1e1eef
round[ 6].is_row     671ef1fd4e2a1e03dfdcbl1ef3d789b30
round[ 6].im_col     0b8a7783417ae3a1f9492dc0c641a7ce
round[ 6].ik_sch     c6deb0ab791e2364a4055f5e568803ab
round[ 7].istart      cd54c7283864c0c55d4c727e90c9a465
round[ 7].is_box     80fd31ee768c1f078d5d1e8a96121dbc
round[ 7].is_row     80121e0776fd1d8a8d8c31bc965d1fee
round[ 7].im_col     4ee1ddf9301d6352c9ad769ef8d20515
round[ 7].ik_sch     dd1b7cdaf28d5c158a49ab1dbbc497cb
round[ 8].istart      93faa123c2903f4743e4dd83431692de
round[ 8].is_box     2214f132a896251664aec94164ff749c
round[ 8].is_row     22ffc916a81474416496f19c64ae2532
round[ 8].im_col     1008ffe53b36ee6af27b42549b8a7bb7
round[ 8].ik_sch     78c4f708318d3cd69655b701bfc093cf
round[ 9].istart      68cc08ed0abbd2bc642ef555244ae878
round[ 9].is_box     f727bf53a3fe7f788cc377eda65cc8c1
round[ 9].is_row     f75c7778a327c8ed8cfefbc1a6c37f53
round[ 9].im_col     7f69ac1ed939ebaac8ece3cb12e159e3
round[ 9].ik_sch     60dcef10299524ce62dbef152f9620cf
round[10].istart     1fb5430ef0accf64aa370cde3d77792c
round[10].is_box     cbd264d717aa5f8c62b2819c8b02af42
round[10].is_row     cb02818c17d2af9c62aa64428bb25fd7
round[10].im_col     cfaf16b2570c18b52e7fef50cab267ae
round[10].ik_sch     4b4ecbdb4d4dcfda5752d7c74949cbde
round[11].istart     84e1dd691a41d76f792d389783fbac70
round[11].is_box     4fe0c9e443f80d06affa76854163aad0
round[11].is_row     4f63760643e0aa85aff8c9d041fa0de4
round[11].im_col     794cf891177bfd1d8a327086f3831b39
round[11].ik_sch     1a1f181d1e1b1c194742c7d74949cbde
round[12].istart     6353e08c0960e104cd70b751bacad0e7
round[12].is_box     0050a0f04090e03080d02070c01060b0
round[12].is_row     00102030405060708090a0b0c0d0e0f0
round[12].ik_sch     000102030405060708090a0b0c0d0e0f
round[12].ioutput    00112233445566778899aabbccddeeff

```

### B.3 AES-256 ( $Nk=8, Nr=14$ )

открытый текст: 00112233445566778899aabbccddeeff

ключ: 000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f

ШИФРОВАНИЕ:

```

round[ 0].input      00112233445566778899aabbccddeeff
round[ 0].k_sch     000102030405060708090a0b0c0d0e0f

```

```
round[ 1].start 00102030405060708090a0b0c0d0e0f0
round[ 1].s_box 63cab7040953d051cd60e0e7ba70e18c
round[ 1].s_row 6353e08c0960e104cd70b751bacad0e7
round[ 1].m_col 5f72641557f5bc92f7be3b291db9f91a
round[ 1].k_sch 101112131415161718191a1b1c1d1e1f
round[ 2].start 4f63760643e0aa85efa7213201a4e705
round[ 2].s_box 84fb386f1ae1ac97df5cfd237c49946b
round[ 2].s_row 84e1fd6b1a5c946fdf4938977cfbac23
round[ 2].m_col bd2a395d2b6ac438d192443e615da195
round[ 2].k_sch a573c29fa176c498a97fce93a572c09c
round[ 3].start 1859fbc28a1c00a078ed8aad42f6109
round[ 3].s_box adcb0f257e9c63e0bc557e951c15ef01
round[ 3].s_row ad9c7e017e55ef25bc150fe01ccb6395
round[ 3].m_col 810dce0cc9db8172b3678c1e88a1b5bd
round[ 3].k_sch 1651a8cd0244beda1a5da4c10640bade
round[ 4].start 975c66c1cb9f3fa8a93a28df8ee10f63
round[ 4].s_box 884a33781fdb75c2d380349e19f876fb
round[ 4].s_row 88db34fb1f807678d3f833c2194a759e
round[ 4].m_col b2822d81abe6fb275faf103a078c0033
round[ 4].k_sch ae87dff00ff11b68a68ed5fb03fc1567
round[ 5].start 1c05f271a417e04ff921c5c104701554
round[ 5].s_box 9c6b89a349f0e18499fda678f2515920
round[ 5].s_row 9cf0a62049fd59a399518984f26be178
round[ 5].m_col aeb65ba974e0f822d73f567bdb64c877
round[ 5].k_sch 6de1f1486fa54f9275f8eb5373b8518d
round[ 6].start c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 6].s_box 2e5bacf8af6ea9e73ac67a34c286ee2d
round[ 6].s_row 2e6e7a2dafc6eef83a86ace7c25ba934
round[ 6].m_col b951c33c02e9bd29ae25cdb1efa08cc7
round[ 6].k_sch c656827fc9a799176f294cec6cd5598b
round[ 7].start 7f074143cb4e243ec10c815d8375d54c
round[ 7].s_box d2c5831a1f2f36b278fe0c4cec9d0329
round[ 7].s_row d22f0c291ffe031a789d83b2ecc5364c
round[ 7].m_col ebb19e1c3ee7c9e87d7535e9ed6b9144
round[ 7].k_sch 3de23a75524775e727bf9eb45407cf39
round[ 8].start d653a4696ca0bc0f5acaab5db96c5e7d
round[ 8].s_box f6ed49f950e06576be74624c565058ff
round[ 8].s_row f6e062ff507458f9be50497656ed654c
round[ 8].m_col 5174c8669da98435a8b3e62ca974a5ea
round[ 8].k_sch 0bdc905fc27b0948ad5245a4c1871c2f
round[ 9].start 5aa858395fd28d7d05e1a38868f3b9c5
round[ 9].s_box bec26a12cfb55dff6bf80ac4450d56a6
round[ 9].s_row beb50aa6cff856126b0d6aff45c25dc4
round[ 9].m_col 0f77ee31d2ccadc05430a83f4ef96ac3
round[ 9].k_sch 45f5a66017b2d387300d4d33640a820a
round[10].start 4a824851c57e7e47643de50c2af3e8c9
round[10].s_box d61352d1a6f3f3a04327d9fee50d9bdd
round[10].s_row d6f3d9dda6279bd1430d52a0e513f3fe
round[10].m_col bd86f0ea748fc4f4630f11c1e9331233
round[10].k_sch 7ccff71cbeb4fe5413e6bbf0d261a7df
```

```

round[11].start      c14907f6ca3b3aa070e9aa313b52b5ec
round[11].s_box      783bc54274e280e0511eacc7e200d5ce
round[11].s_row      78e2acce741ed5425100c5e0e23b80c7
round[11].m_col      af8690415d6e1dd387e5fbbedd5c89013
round[11].k_sch      f01afafee7a82979d7a5644ab3afe640
round[12].start      5f9c6abfbac634aa50409fa766677653
round[12].s_box      cfde0208f4b418ac5309db5c338538ed
round[12].s_row      cfb4dbedf4093808538502ac33de185c
round[12].m_col      7427fae4d8a695269ce83d315be0392b
round[12].k_sch      2541fe719bf500258813bbd55a721c0a
round[13].start      516604954353950314fb86e401922521
round[13].s_box      d133f22a1aed2a7bfa0f44697c4f3ffd
round[13].s_row      d1ed44fd1a0f3f2afa4ff27b7c332a69
round[13].m_col      2c21a820306f154ab712c75eee0da04f
round[13].k_sch      4e5a6699a9f24fe07e572baacdf8cdea
round[14].start      627bceb9999d5aaac945ecf423f56da5
round[14].s_box      aa218b56ee5ebeacdd6ecebf26e63c06
round[14].s_row      aa5ece06ee6e3c56dde68bac2621bebf
round[14].k_sch      24fc79ccb0979e9371ac23c6d68de36
round[14].output     8ea2b7ca516745bfeafc49904b496089

```

#### РАСШИФРОВАНИЕ:

```

round[ 0].iinput     8ea2b7ca516745bfeafc49904b496089
round[ 0].ik_sch     24fc79ccb0979e9371ac23c6d68de36
round[ 1].istart     aa5ece06ee6e3c56dde68bac2621bebf
round[ 1].is_row     aa218b56ee5ebeacdd6ecebf26e63c06
round[ 1].is_box     627bceb9999d5aaac945ecf423f56da5
round[ 1].ik_sch     4e5a6699a9f24fe07e572baacdf8cdea
round[ 1].ik_add     2c21a820306f154ab712c75eee0da04f
round[ 2].istart     d1ed44fd1a0f3f2afa4ff27b7c332a69
round[ 2].is_row     d133f22a1aed2a7bfa0f44697c4f3ffd
round[ 2].is_box     516604954353950314fb86e401922521
round[ 2].ik_sch     2541fe719bf500258813bbd55a721c0a
round[ 2].ik_add     7427fae4d8a695269ce83d315be0392b
round[ 3].istart     cfb4dbedf4093808538502ac33de185c
round[ 3].is_row     cfde0208f4b418ac5309db5c338538ed
round[ 3].is_box     5f9c6abfbac634aa50409fa766677653
round[ 3].ik_sch     f01afafee7a82979d7a5644ab3afe640
round[ 3].ik_add     af8690415d6e1dd387e5fbbedd5c89013
round[ 4].istart     78e2acce741ed5425100c5e0e23b80c7
round[ 4].is_row     783bc54274e280e0511eacc7e200d5ce
round[ 4].is_box     c14907f6ca3b3aa070e9aa313b52b5ec
round[ 4].ik_sch     7ccff71cbeb4fe5413e6bbf0d261a7df
round[ 4].ik_add     bd86f0ea748fc4f4630f11c1e9331233
round[ 5].istart     d6f3d9dda6279bd1430d52a0e513f3fe
round[ 5].is_row     d61352d1a6f3f3a04327d9fee50d9bdd
round[ 5].is_box     4a824851c57e7e47643de50c2af3e8c9
round[ 5].ik_sch     45f5a66017b2d387300d4d33640a820a
round[ 5].ik_add     0f77ee31d2ccadc05430a83f4ef96ac3

```

```
round[ 5].ik_sch 45f5a66017b2d387300d4d33640a820a
round[ 5].ik_add 0f77ee31d2ccadc05430a83f4ef96ac3
round[ 6].istart beb50aa6cff856126b0d6aff45c25dc4
round[ 6].is_row bec26a12cfb55dff6bf80ac4450d56a6
round[ 6].is_box 5aa858395fd28d7d05e1a38868f3b9c5
round[ 6].ik_sch 0bdc905fc27b0948ad5245a4c1871c2f
round[ 6].ik_add 5174c8669da98435a8b3e62ca974a5ea
round[ 7].istart f6e062ff507458f9be50497656ed654c
round[ 7].is_row f6ed49f950e06576be74624c565058ff
round[ 7].is_box d653a4696ca0bc0f5acaab5db96c5e7d
round[ 7].ik_sch 3de23a75524775e727bf9eb45407cf39
round[ 7].ik_add ebb19e1c3ee7c9e87d7535e9ed6b9144
round[ 8].istart d22f0c291ffe031a789d83b2ecc5364c
round[ 8].is_row d2c5831a1f2f36b278fe0c4cec9d0329
round[ 8].is_box 7f074143cb4e243ec10c815d8375d54c
round[ 8].ik_sch c656827fc9a799176f294cec6cd5598b
round[ 8].ik_add b951c33c02e9bd29ae25cdb1efa08cc7
round[ 9].istart 2e6e7a2dafc6eef83a86ace7c25ba934
round[ 9].is_row 2e5bacf8af6ea9e73ac67a34c286ee2d
round[ 9].is_box c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 9].ik_sch 6de1f1486fa54f9275f8eb5373b8518d
round[ 9].ik_add aeb65ba974e0f822d73f567bdb64c877
round[10].istart 9cf0a62049fd59a399518984f26be178
round[10].is_row 9c6b89a349f0e18499fda678f2515920
round[10].is_box 1c05f271a417e04ff921c5c104701554
round[10].ik_sch ae87dff00ff11b68a68ed5fb03fc1567
round[10].ik_add b2822d81abe6fb275faf103a078c0033
round[11].istart 88db34fb1f807678d3f833c2194a759e
round[11].is_row 884a33781fdb75c2d380349e19f876fb
round[11].is_box 975c66c1cb9f3fa8a93a28df8ee10f63
round[11].ik_sch 1651a8cd0244beda1a5da4c10640bade
round[11].ik_add 810dce0cc9db8172b3678c1e88a1b5bd
round[12].istart ad9c7e017e55ef25bc150fe01ccb6395
round[12].is_row adcb0f257e9c63e0bc557e951c15ef01
round[12].is_box 1859fbc28a1c00a078ed8aad42f6109
round[12].ik_sch a573c29fa176c498a97fce93a572c09c
round[12].ik_add bd2a395d2b6ac438d192443e615da195
round[13].istart 84e1fd6b1a5c946fdf4938977cfbac23
round[13].is_row 84fb386f1ae1ac97df5cfd237c49946b
round[13].is_box 4f63760643e0aa85efa7213201a4e705
round[13].ik_sch 101112131415161718191a1b1c1d1e1f
round[13].ik_add 5f72641557f5bc92f7be3b291db9f91a
round[14].istart 6353e08c0960e104cd70b751bacad0e7
round[14].is_row 63cab7040953d051cd60e0e7ba70e18c
round[14].is_box 00102030405060708090a0b0c0d0e0f0
round[14].ik_sch 000102030405060708090a0b0c0d0e0f
round[14].ioutput 00112233445566778899aabbccddeeff
```

ЭКВИВАЛЕНТНАЯ ПРОЦЕДУРА  
РАСШИФРОВАНИЯ:

```
round[ 0] .iinput      8ea2b7ca516745bfeafc49904b496089
round[ 0] .ik_sch     24fc79ccb0979e9371ac23c6d68de36
round[ 1] .istart     aa5ece06ee6e3c56dde68bac2621bebf
round[ 1] .is_box     629deca599456db9c9f5ceaa237b5af4
round[ 1] .is_row     627bceb9999d5aaac945ecf423f56da5
round[ 1] .im_col     e51c9502a5c1950506a61024596b2b07
round[ 1] .ik_sch     34f1d1ffbfceaa2ffce9e25f2558016e
round[ 2] .istart     d1ed44fd1a0f3f2afa4ff27b7c332a69
round[ 2] .is_box     5153862143fb259514920403016695e4
round[ 2] .is_row     516604954353950314fb86e401922521
round[ 2] .im_col     91a29306cc450d0226f4b5eaef5efed8
round[ 2] .ik_sch     5e1648eb384c350a7571b746dc80e684
round[ 3] .istart     cfb4dbedf4093808538502ac33de185c
round[ 3] .is_box     5fc69f53ba4076bf50676aaa669c34a7
round[ 3] .is_row     5f9c6abfbac634aa50409fa766677653
round[ 3] .im_col     b041a94eff21ae9212278d903b8a63f6
round[ 3] .ik_sch     c8a305808b3f7bd043274870d9b1e331
round[ 4] .istart     78e2acce741ed5425100c5e0e23b80c7
round[ 4] .is_box     c13baaeccae9b5f6705207a03b493a31
round[ 4] .is_row     c14907f6ca3b3aa070e9aa313b52b5ec
round[ 4] .im_col     638357cec07de6300e30d0ec4ce2a23c
round[ 4] .ik_sch     b5708e13665a7de14d3d824ca9f151c2
round[ 5] .istart     d6f3d9dda6279bd1430d52a0e513f3fe
round[ 5] .is_box     4a7ee5c9c53de85164f348472a827e0c
round[ 5] .is_row     4a824851c57e7e47643de50c2af3e8c9
round[ 5] .im_col     ca6f71058c642842a315595fdf54f685
round[ 5] .ik_sch     74da7ba3439c7e50c81833a09a96ab41
round[ 6] .istart     beb50aa6cff856126b0d6aff45c25dc4
round[ 6] .is_box     5ad2a3c55fe1b93905f3587d68a88d88
round[ 6] .is_row     5aa858395fd28d7d05e1a38868f3b9c5
round[ 6] .im_col     ca46f5ea835eab0b9537b6dbb221b6c2
round[ 6] .ik_sch     3ca69715d32af3f22b67ffade4ccd38e
round[ 7] .istart     f6e062ff507458f9be50497656ed654c
round[ 7] .is_box     d6a0ab7d6cca5e695a6ca40fb953bc5d
round[ 7] .is_row     d653a4696ca0bc0f5acaab5db96c5e7d
round[ 7] .im_col     2a70c8da28b806e9f319ce42be4baead
round[ 7] .ik_sch     f85fc4f3374605f38b844df0528e98e1
round[ 8] .istart     d22f0c291ffe031a789d83b2ecc5364c
round[ 8] .is_box     7f4e814ccb0cd543c175413e8307245d
round[ 8] .is_row     7f074143cb4e243ec10c815d8375d54c
round[ 8] .im_col     f0073ab7404a8a1fc2cba0b80df08517
round[ 8] .ik_sch     de69409aef8c64e7f84d0c5fcfab2c23
```

```

round[ 9] .istart      2e6e7a2dafc6eef83a86ace7c25ba934
round[ 9] .is_box     c345bdfa1bc799e1a2dcaab0a857b728
round[ 9] .is_row     c357aae11b45b7b0a2c7bd28a8dc99fa
round[ 9] .im_col     3225fe3686e498a32593c1872b613469
round[ 9] .ik_sch     aed55816cf19c100bcc24803d90ad511
round[10] .istart     9cf0a62049fd59a399518984f26be178
round[10] .is_box     1c17c554a4211571f970f24f0405e0c1
round[10] .is_row     1c05f271a417e04ff921c5c104701554
round[10] .im_col     9d1d5c462e655205c4395b7a2eac55e2
round[10] .ik_sch     15c668bd31e5247d17c168b837e6207c
round[11] .istart     88db34fb1f807678d3f833c2194a759e
round[11] .is_box     979f2863cb3a0fc1a9e166a88e5c3fdf
round[11] .is_row     975c66c1cb9f3fa8a93a28df8ee10f63
round[11] .im_col     d24bfb0e1f997633cfce86e37903fe87
round[11] .ik_sch     7fd7850f61cc991673db890365c89d12
round[12] .istart     ad9c7e017e55ef25bc150fe01ccb6395
round[12] .is_box     181c8a098aed61c2782ffba0c45900ad
round[12] .is_row     1859fbc28a1c00a078ed8aad42f6109
round[12] .im_col     aec9bda23e7fd8aff96d74525cdce4e7
round[12] .ik_sch     2a2840c924234cc026244cc5202748c4
round[13] .istart     84e1fd6b1a5c946fdf4938977cfbac23
round[13] .is_box     4fe0210543a7e706efa476850163aa32
round[13] .is_row     4f63760643e0aa85efa7213201a4e705
round[13] .im_col     794cf891177bfd1ddf67a744acd9c4f6
round[13] .ik_sch     1a1f181d1e1b1c191217101516131411
round[14] .istart     6353e08c0960e104cd70b751bacad0e7
round[14] .is_box     0050a0f04090e03080d02070c01060b0
round[14] .is_row     00102030405060708090a0b0c0d0e0f0
round[14] .ik_sch     000102030405060708090a0b0c0d0e0f
round[14] .ioutput    00112233445566778899aabbccddeeff

```

## Приложение Г – ссылки

- [1] - страница, посвящённая алгоритму AES – <http://www.nist.gov/CriptoToolkit>
- [2] – реестр компьютерных объектов защиты (CSOR) – <http://csrc.nist.gov/csor/>
- [3] – J. Daemen и V. Rijmen, *AES Proposal: Rijndael*, AES Algorithm Submission, 3 сентября 1999 г., доступно по ссылке [1]
- [4] - J. Daemen и V. Rijmen, *The block cipher Rijndael*, Smart card reseach and Applications, LNCS 1820, Springer-Verlag, страницы 288-296.
- [5] – домашняя страница В. Glanman’a, посвящённая алгоритму AES – [http://fp.gladman.plus.com/cryptography\\_technology](http://fp.gladman.plus.com/cryptography_technology)
- [6] – A. Lee, NIST Spesial Publication 800-21, *Guideline for Implementing Cryptography in the Federal Government*, National Institute of Standards and Technology, ноябрь 1999
- [7] – A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997, станицы 81-83.
- [8] – J. Nechvatal и др., *Report on the Development of the Advanced Encryption Standard (AES)*, National Institute of Standards and Technology, 2 октября 2000 г., доступно по ссылке [1]