ПРАКТИЧЕСКАЯ КРИПТОЛОГИЯ
ЛЕКЦИЯ 9
Специальность: 6.170101 – Бсіт
Лектор: Сушко С.А.

§15. ОБЩЕЕ ОПИСАНИЕ КРИПТОАЛГОРИТМА AES

AES – аббревиатура от Advanced Encryption Standard (перевод с англ. усовершенствованный стандарт шифрования).

После открытия в 1997 г. Национальным Институтом Стандартов и Технологии США (NIST) программы по разработке AES состоялись 3 этапа международного конкурса. Новый стандарт должен был иметь

- стойкость не меньше, чем у 3DES;
- скорость шифрования больше скорости 3DES;
- прозрачную структуру;
- эффективную реализацию на платформе Pentium Pro;
- эффективную аппаратную реализацию.

Победителем конкурса стали бельгийцы Йоан Дамен и Винсент Реймен с алгоритмом RIJNDAEL (читается Рейн-дал от первых букв авторов). Стандарт начал действовать с 2002 р.

AES – симметричный итеративный блоковый алгоритм;

AES – не шифр Фейстеля, базируется на принципах новой сети подстановок-перестановок. Имеет новую архитектуру SQUARE (КВАДРАТ), для которой характерно: 1) представление шифруемого блока в виде двумерного байтового массива; 2) шифрование за один раунд всего блока данных (***байт-ориентированная структура***); 3) выполнение криптографических преобразований, как над отдельными байтами массива, так и над его строками и столбцами. Это обеспечивает диффузию данных одновременно в двух направлениях - по строкам и по столбцам. Архитектура SQUARE присуща, кроме шифра AES(RIJNDAEL), шифрам SQUARE (его название и дало имя всей архитектуре), CRYPTON (один из кандидатов на AES). Второе место в конкурсе AES занял другой SP-шифр, SERPENT. По-видимому, SP-сети и, в частности, архитектура SQUARE, в ближайшем будущем станут безраздельно доминировать.

Общие характеристики AES

- AES зашифровывает и расшифровывает 128-битовые блоки данных.
- AES позволяет использовать три различных ключа длиной 128, 192 или 256 бит (в зависимости от длины ключа версии шифра обозначают AES-128, AES-192 или AES-256).
- От размера ключа зависит число раундов шифрования:
  длина 128 бит – 10 раундов;
  длина 192 бита – 12 раундов;
  длина 256 бит – 14 раундов.
- Все раунды, кроме последнего, идентичны.

§16. ПРЕДСТАВЛЕНИЕ ДАННЫХ В КРИПТОАЛГОРИТМЕ AES

Напоминание: 1 байт = 8 битов, 128 битов=16×8 битов=16 байт.

Основным элементом, которым оперирует алгоритм AES, является байт – последовательность 8 бит, обрабатываемых как единое целое. Для формирования байтов 128 битов блока открытого текста, выходного блока шифротекста и ключа шифра делятся на группы из 8-ми рядом стоящих бит так, чтобы в целом получился массив байт. Ниже представлена принятая нумерация бит в пределах каждого байта:

№ бита на входе	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
№ байта	0								1								2								…
№ 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	…

Задавать значение байта удобно в шестнадцатеричной системе исчисления. Для этого байт делится на две группы из 4-х бит: группа старших бит в байте представляется первым шестнадцатеричным символом, а группа младших бит – вторым. Например, для байта 10101100 получим

$$10101100 = 1010\ \ 1100 = AC.$$

Обозначим

$in_0, in_1, \ldots, in_{15}$ – 16 байт блока открытого текста;

$k_0, k_1, \ldots, k_{15}$ – 16 байт ключа шифра;

– 16 байт блока шифротекста.

Входными данными для операций шифрования есть массив из 16 байт $in_0, in_1, \ldots, in_{15}$. Перед началом шифрования байты этого массива размещаются последовательно в <u>столбцы</u> матрицы $\mathrm{Input}Block$ (сверху вниз). Внутри алгоритма операции выполняются над матрицей байт, называемой ***матрицей состояний*** $State$ или просто ***состоянием***. Конечное значение матрицы состояния $\mathrm{Output}Block$ является выходом алгоритма и преобразуется в последовательность байтов шифротекста $out_0, out_1, \ldots, out_{15}$. Аналогично в столбцы матрицы $\mathrm{Input}Key$ попадают и 16 байтов $k_0, k_1, ..., k_{15}$ ключа шифра. Размерность всех матриц – 4×4.

Схематически такое представление данных выглядит так:

Входные байты		Выходные байты
$\mathrm{Input}Block$	$State$	$\mathrm{Output}Block$

in_0	in_4	in_0	in_0
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

$\rightarrow$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$=$

out_0	out_4	out_0	out_0
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Байты ключа
Input*Key*

k_0	k_4	k_0	k_0
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Четыре байта в каждом столбце матрицы состояний или ключа можно рассматривать как одно 32-х битовое ***слово***. Поэтому матрица состояний – это массив из 4 слов w_0, w_1, w_2, w_3,
где

$$w_0 = s_{0,0}\, s_{1,0}\, s_{2,0}\, s_{3,0};$$

$$w_1 = s_{0,1}\, s_{1,1}\, s_{2,1}\, s_{3,1};$$

$$w_2 = s_{0,2}\, s_{1,2}\, s_{2,2}\, s_{3,2};$$

$$w_3 = s_{0,3}\, s_{1,3}\, s_{2,3}\, s_{3,3}.$$

Матрица, поступающая на вход каждого раунда называется матрицей Input*State*, а на выходе раунда образуется матрица Output*State*. Очевидно, на входе первого раунда Input*State*= Input*Block* , а на выходе последнего раунда Output*State*= Output*Block*.

Например, представим в виде матрицы Input*Block* текст
СКЛАДНІСТЬЗАДАЧІ = (21 14 15 00 05 17 11 21 22 30 09 00 05 00 27 11)$_{10}$=
= (15 0E 0F 00 05 11 0B 15 16 1E 09 00 05 00 1B 0B)$_{16}$

$$\text{Input}Block = \begin{pmatrix} 15 & 05 & 16 & 05 \\ 0E & 11 & 1E & 00 \\ 0F & 0B & 09 & 1B \\ 00 & 15 & 00 & 0B \end{pmatrix}.$$

§17. ОБЩАЯ СТРУКТУРА AES

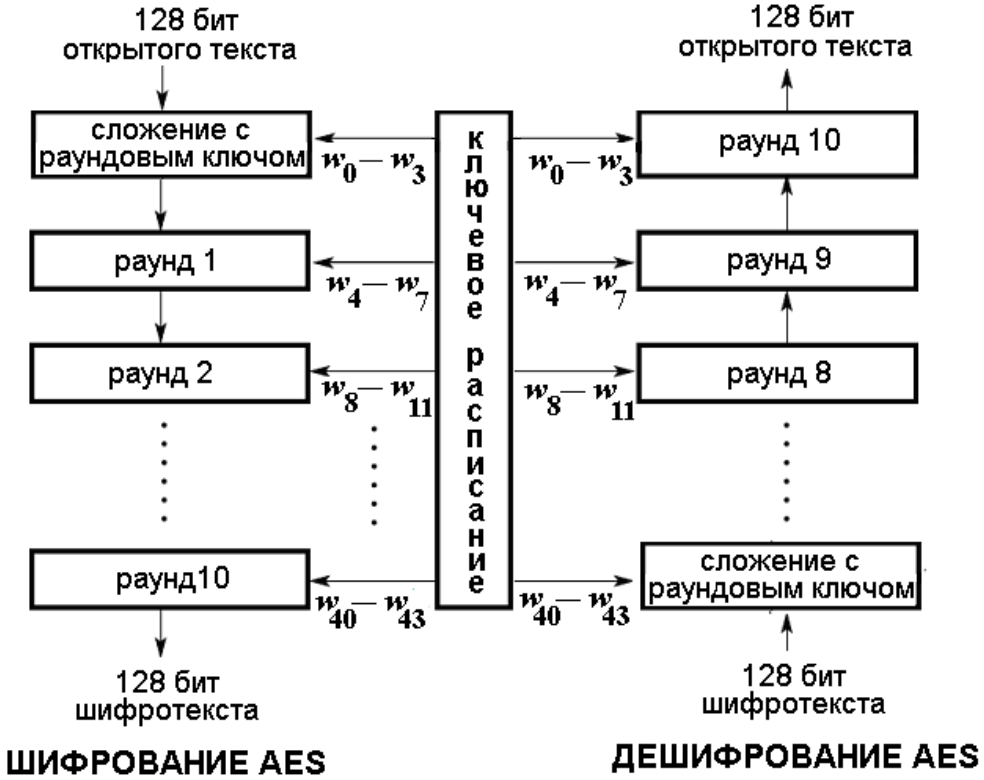
Как отмечалось, в рассматриваемой версии алгоритма AES-128 ключ шифра состоит из 128 битов, поделенных на 16 байтов $k_0, k_1, ..., k_{15}$, и записывается в столбцы матрицы Input*Key*. Каждый

столбец матрицы $\mathrm{Input}Key$ образует слово, т.е. фактически ключ шифра – это четыре слова w_0, w_1, w_2, w_3, где $w_0 = k_0 k_1 k_2 k_3$, $w_1 = k_4 k_5 k_6 k_7$ и т.д.

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Из этих слов с помощью специального алгоритма (о нем позже) образуется последовательность из 44 слов: $w_0, w_1, w_2, ..., w_{43}$ (каждое слово по 32 бита). На каждый раунд шифрования подаются по четыре слова этой последовательности. Они и будут играть роль раундового ключа.

Схема преобразования данных показана на рисунке.



Перед первым раундом выполняется операция AddRound*Key* (суммирование по модулю 2 с начальным ключом шифра). Преобразования, выполненные в одном раунде, обозначают

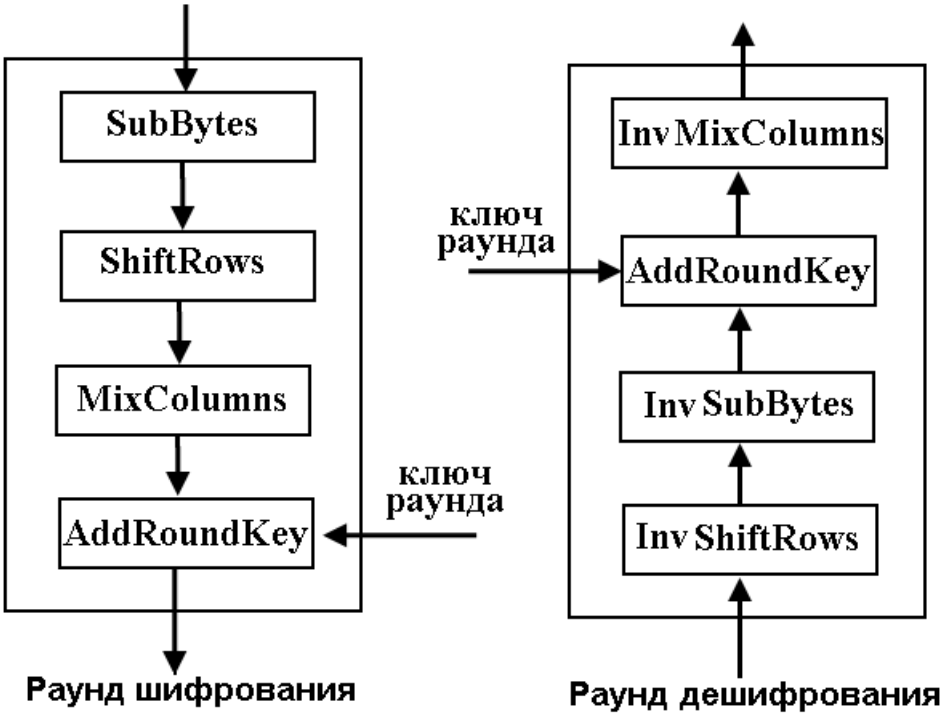
$$\mathrm{Round}(State,\ RoundKey),$$

где переменная *State* – матрица, описывающая данные на входе раунда и на его выходе после шифрования; переменная *RoundKey* – матрица, содержащая раундовый ключ.

Раунд состоит из 4 различных преобразований:

- SubBytes – побайтовая подстановка в *S*-боксе с фиксированной таблицей замен;
- ShiftRows – побайтовый сдвиг строк матрицы *State* на различное количество байт;
- MixColumns – перемешивание байт в столбцах;
- AddRoundKey – сложение с раундовым ключом (операция *XOR*).

Последний раунд несколько отличается от предыдущих тем, что не задействует функцию MixColumns.



При дешифровании в каждом раунде выполняются обратные операции: InvShiftRows, InvSubBytes, AddRoundKey и InvMixColumns (в обозначениях перед названием функции появляется приставка Inv). Порядок выполнения операций при шифровании и дешифровании различен, причины чего будут ясны после детального рассмотрения каждого преобразования.

§18. МАТЕМАТИЧЕСКИЕ ОСНОВЫ ШИФРА AES

Операции в поле $GF(2^8)$**.** Для описания алгоритма используется конечное поле Галуа $GF(2^8)$, построенное как расширение поля $GF(2)=\{0,1\}$ по модулю неприводимого многочлена $m(x)=x^8+x^4+x^3+x+1.$ Элементами поля $GF(2^8)$ являются многочлены вида

$$b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0,$$

степень которых меньше 8, а коэффициенты $b_7,b_6,...,b_0\in\{0,1\}$. Операции в поле выполняются по модулю $m(x)$. Всего в поле $GF(2^8)$ насчитывается $2^8=256$ многочленов.

Представление двоичного числа $b_7b_6b_5b_4b_3b_2b_1b_0$ в виде многочлена с коэффициентами $b_7,b_6,...,b_0$ позволяет интерпретировать байт как битовый многочлен в конечном поле $GF(2^8)$:

$$b(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0. \quad (1)$$

Например, байт 63 задает последовательность битов 01100011 и определяет конкретный элемент поля

$$01100011\leftrightarrow x^6+x^5+x+1.$$

Рассмотрим основные математические операции в поле $GF(2^8)$.

1. ***Сложение байт*** можно выполнить любым из трех способов:
   - представить байты битовыми многочленами и сложить их по обычному правилу суммирования многочленов с последующим приведением коэффициентов суммы по модулю 2 (операция XOR над коэффициентами);
   - суммировать по модулю 2 соответствующие биты в байтах;
   - сложить байты в шестнадцатеричной системе исчисления.

Например, следующие три записи эквивалентны:

- представление в виде многочленов
  $$(x^6+x^4+x^2+x+1)+(x^7+x+1)=x^7+x^6+x^4+x^2;$$
- битовое представление
  $$\{01010111\}\oplus\{10000011\}=\{11010100\};$$
- шестнадцатеричное представление
  $$\{57\}\oplus\{83\}=\{D4\}.$$

2. ***Умножение байт*** выполняется с помощью представления их многочленами и перемножения по обычным алгебраическим правилам. Полученное произведение необходимо привести по модулю многочлена

$m(x) = x^8 + x^4 + x^3 + x + 1$ (результат приведения равен остатку от деления произведения на $m(x)$).

Перемножение многочленов в поле можно упростить, введя операцию умножения битового многочлена (1) на x:

$$x(b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0) =$$
$$= b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x.$$

3. Для любого ненулевого битового многочлена $b(x)$ в поле $GF(2^8)$ существует многочлен $b^{-1}(x)$, ***обратный*** к нему по умножению, т.е. $b(x)b^{-1}(x) \equiv 1 \bmod m(x)$. Для нахождения обратного элемента используют расширенный алгоритм Эвклида, с помощью которого находят такие многочлены $a(x)$ и $c(x)$, что $a(x)b(x) + c(x)m(x) = 1$. Следовательно, $b^{-1}(x) \equiv a(x) \bmod m(x)$.

Многочлены с коэффициентами, принадлежащими полю $GF(2^8)$. Многочлены третьей степени с коэффициентами из конечного поля $a_i \in GF(2^8)$ имеют вид:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0. \qquad (2)$$

Таким образом, в этих многочленах в роли коэффициентов при неизвестных задействованы байты вместо бит. Далее многочлены (2) будем представлять в форме слова $[a_0, a_1, a_2, a_3]$. В стандарте AES при умножении многочленов вида (2) используется приведение по модулю другого многочлена $x^4 + 1$.

Для изучения арифметики рассматриваемых многочленов введем дополнительно многочлен $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$, где $b_i \in GF(2^8)$. Тогда

1. ***Сложение.***

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

2. ***Умножение.***

Это более сложная операция. Пусть, мы перемножаем два многочлена $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ и $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$. Результатом умножения будет многочлен

$$c(x) = a(x) \cdot b(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0,$$

где
$$c_0 = a_0b_0;$$
$$c_1 = a_1b_0 \oplus a_0b_1;$$
$$c_2 = a_2b_0 \oplus a_1b_1 \oplus a_0b_2;$$

$$c_3 = a_3b_0 \oplus a_2b_1 \oplus a_1b_2 \oplus a_0b_3;$$
$$c_4 = a_3b_1 \oplus a_2b_2 \oplus a_1b_3;$$
$$c_5 = a_3b_2 \oplus a_2b_3;$$
$$c_6 = a_3b_3.$$

Чтобы результат можно было представить четырехбайтовым словом, необходимо взять результат по модулю многочлена степени не более 4. Авторы шифра выбрали для этой цели многочлен x^4+1, для которого справедливо $x^i \bmod (x^4+1) \equiv x^{i \bmod 4}$.

Поэтому в произведении коэффициенты при степенях x^i, $i = 0,1,2,3$, равны сумме произведений $a_j b_k$ по индексам, для которых $j + k = i (\bmod 4)$, $j,k = 0,1,2,3$.

Таким образом, после приведения по модулю x^4+1 получим

$$d(x) = a(x) \cdot b(x) = d_3x^3 + d_2x^2 + d_1x + d_0x,$$

где $d_0 = a_0b_0 \oplus a_3b_1 \oplus a_2b_2 \oplus a_1b_3;$
$d_1 = a_1b_0 \oplus a_0b_1 \oplus a_3b_2 \oplus a_2b_3;$
$d_2 = a_2b_0 \oplus a_1b_1 \oplus a_0b_2 \oplus a_3b_3;$
$d_3 = a_3b_0 \oplus a_2b_1 \oplus a_1b_2 \oplus a_0b_3.$

или в матричной форме:

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

§19. РАУНДОВОЕ ПРЕОБРАЗОВАНИЕ AES

Рассмотрим подробнее преобразования раунда шифрования.

1. **Операция** SubBytes**.**

Операция выполняет нелинейную замену байтов, выполняемую независимо с каждым байтом матрицы *State*. Замена обратима и построена путем комбинации двух преобразований над входным байтом:

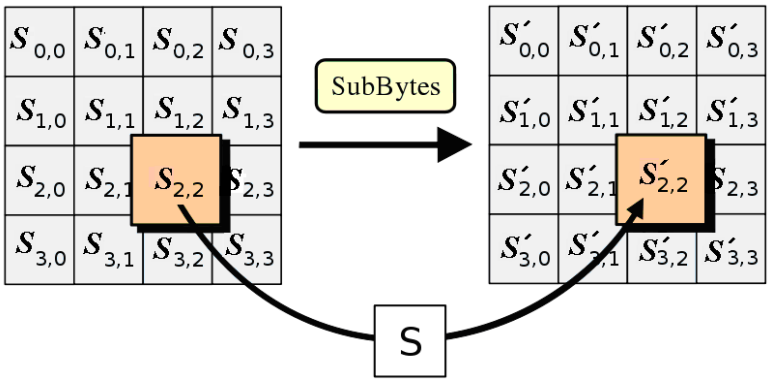
- нахождение обратного (инвертированного) элемента относительно умножения в поле $GF(2^8)$ (считается, что нулевой байт $\{00\}$ переходит сам в себя);
- выполнение некого аффинного преобразования: умножение инвертированного байта на многочлен

$a(x) = x^4 + x^3 + x^2 + x + 1$ и суммирование с многочленом $b(x) = x^6 + x^5 + x + 1$ в поле $F_2[x]/x^8 + 1$. Заметим, что $a^{-1}(x) = x^6 + x^3 + x$ и $a^{-1}(x)b(x) = x^2 + 1$

В матричной форме процедура SubBytes записывается как

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

где через x обозначены входные биты, а через y – выходные. Если на вход функции попадает нулевой байт, то результатом замены будет число $y = b$. Процесс замены байтов с помощью таблицы подстановки иллюстрирует рисунок. Нелинейность преобразования обусловлена нелинейностью инверсии x^{-1}, а обратимость – обратимостью матрицы.



Созданную на основе этой операции специальную таблицу замен байтов в шестнадцатеричной системе называют ***S*-боксом**.

	Таблица преобразования SubBytes															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	FO	AD	D4	A2	AF	9C	A4	72	CO
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	AO	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	DO	EF	AA	FB	43	4D	33	85	45	F9	02	F7	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	DE
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	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	IF	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	OE	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	OD	BF	E6	42	68	41	99	2D	OF	BO	54	BB	16

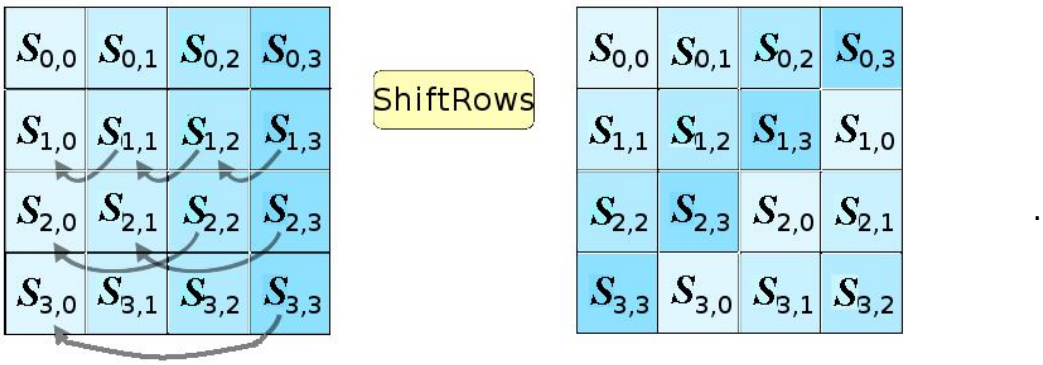
Например, если $s_{1,1} = \{8A\}$, то результат замены этого байта следует искать на пересечении строки с индексом 8 и столбца с индексом A, т.е. SubBytes$(8A)=\{7E\}$.

2. **Операция** ShiftRows.

Операция применяется к строкам матрицы $State$ – ее первая строка неподвижна, а элементы нижних трех строк циклически сдвигаются вправо на 1, 2 и 3 байта соответственно.

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}.$$

По сути это перестановка элементов матрицы, в которой участвуют только элементы строк, поэтому преобразование обратимо.



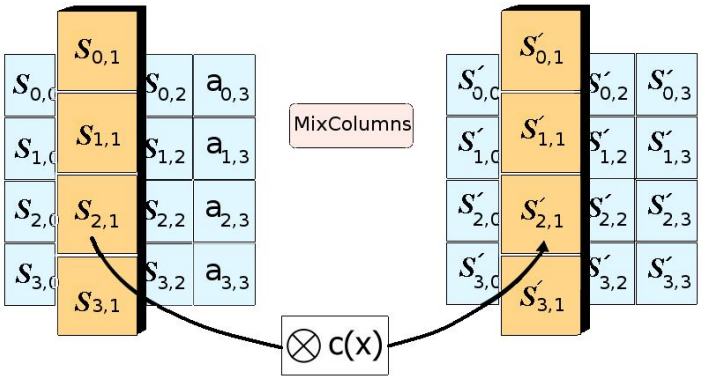
3. **Операция** MixColumns.

С помощью этой операции выполняется перемешивание байтов в столбцах матрицы $State$. Каждый столбец этой матрицы принимается за многочлен над полем $GF(2^8)$ и умножается на фиксированный многочлен

$$c(x) = c_3x^3 + c_2x^2 + cx + c_0 = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

по модулю многочлена x^4+1 (напомним, все коэффициенты многочленов над полем $GF(2^8)$– байты). Как показано выше, такую операцию можно записать в матричном виде как

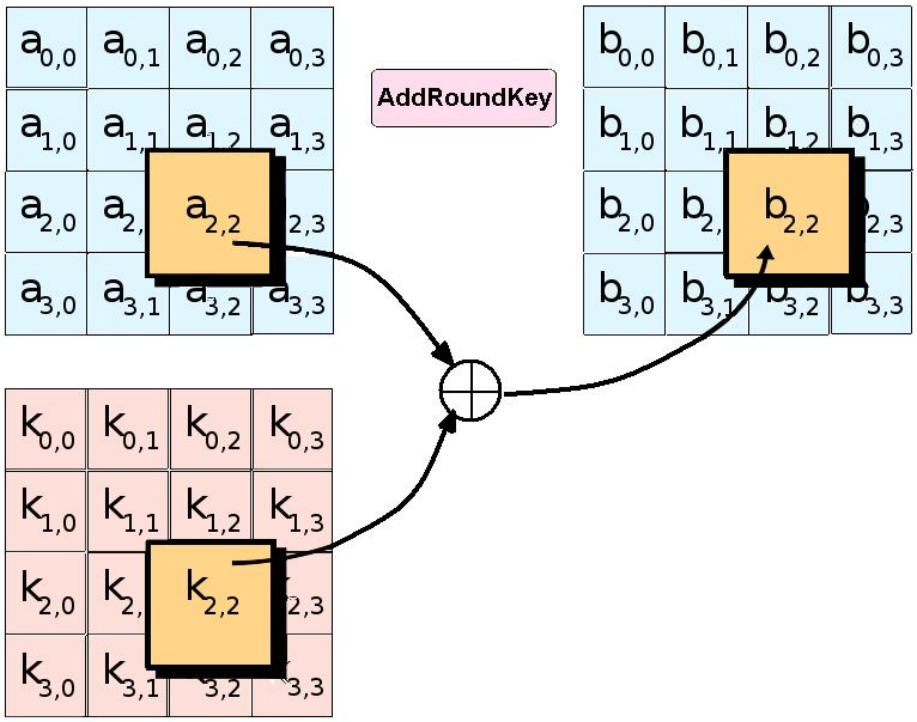
$$\begin{pmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} s_0' \\ s_1' \\ s_2' \\ s_3' \end{pmatrix}.$$



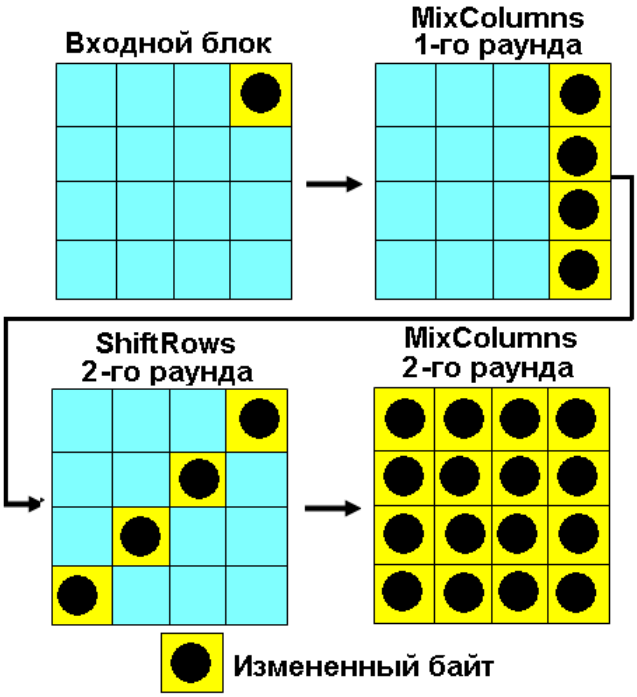
Многочлен $c(x)$ – взаимно простой с многочленом x^4+1 над полем $GF(2)$, поэтому в поле существует обратный многочлен $c^{-1}(x)(\mathrm{mod}\, x^4+1) \Rightarrow$ матрица в этой формуле обратима.

4. **Операция** AddRoundKey**.**

Функция AddRoundKey($State, RoundKey$) побитово складывает элементы переменной $RoundKey$ и элементы переменной $State$ по принципу: i-й столбец данных $(i=0,1,2,3)$ складывается с определенным 4-байтовым фрагментом расширенного ключа $W[4r+1]$, где r – номер поточного раунда алгоритма. При шифровании первое сложение ключа раунда происходит до первого выполнения операции SubBytes.



Следующий рисунок демонстрирует свойства рассеивания и перемешивания информации в ходе шифрование алгоритмом AES. Видно, что два раунда обеспечивают полное рассеивание и перемешивание информации. Достигается это за счет использования функций ShiftRows и MixColumns. Операция SubBytes придает шифрованию стойкость против дифференциального криптоанализа, а операция AddRoundKey обеспечивает необходимую секретную случайность.



§20. КЛЮЧЕВОЕ РАСПИСАНИЕ AES

Раундовые ключи вырабатываются из ключа шифра K с помощью процедуры ***расширения ключа***, в результате чего формируется массив раундовых ключей, из которого затем непосредственно выбирается необходимый раундовый ключ.

Каждый раундовый ключ имеет длину 128 бит (или 4 четырехбайтовых слова $w_i, w_{i+1}, w_{i+2}, w_{i+3}$, а длина в битах всех раундовых ключей равна 128 бит·(10 раундов+1) = 1408 бит (или 44 четырехбайтовых слова $w_0, w_1, w_2, ..., w_{42}, w_{43}$). Первые четыре слова w_0, w_1, w_2, w_3 в ключевом массиве заполнены ключом шифра, из остальных выработанных 40 слов выбираются по 4 слова для ключа раунда. Выбор слов прост: первые четыре слова (они совпадают с ключом шифра) являются ключом с номером 0, следующие четыре слова w_4, w_5, w_6, w_7 – раундовым ключом для первого полного раунда и т.д.

Новые слова $w_{i+4}, w_{i+5}, w_{i+6}, w_{i+7}$ следующего раундового ключа определяются из слов $w_i, w_{i+1}, w_{i+2}, w_{i+3}$ предыдущего ключа на основе уравнений:

$$w_{i+5} = w_{i+4} \oplus w_{i+1};$$
$$w_{i+6} = w_{i+5} \oplus w_{i+2};$$
$$w_{i+7} = w_{i+6} \oplus w_{i+3}.$$

Первое слово w_{i+4} в каждом раундовом ключе изменяется по-другому:

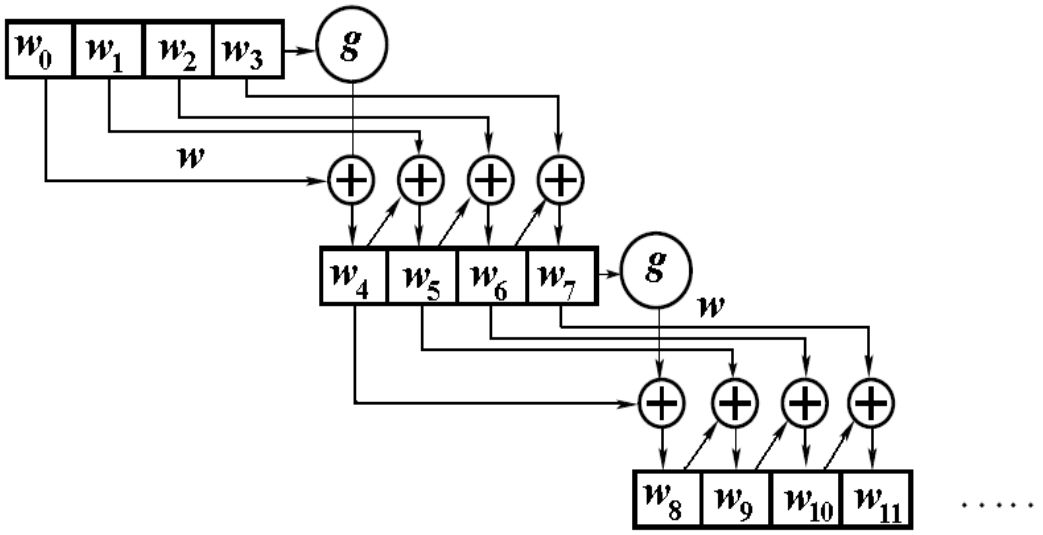
$$w_{i+4} = w_i \oplus g(w_{i+3}),$$

Здесь действие функции g сводится к последовательному выполнению трех шагов, отображающих слово в слово:

1. 1^0 циклический сдвиг четырехбайтового слова влево на один байт (операция $\mathrm{RotWord}$);
2. 2^0 замена каждого байта слова, полученного на шаге 1^0, в соответствии с таблицей $\mathrm{SubBytes}$, используемой при шифровании (операция $\mathrm{SubWord}$);
3. 3^0 суммирование по $\mathrm{mod}\,2$ байтов, полученных на шаге 2^0, с раундовой постоянной $R_{con}[i] = (RC[i], 0, 0, 0)$, несекретной и уникальной для каждого раундового ключа K_i. Три самые правые байты этой константы – нулевые, а ненулевой левый байт меняется по известному закону рекурсии: $RC[1] = 1$, $RC[i] = 2 \cdot RC[i-1]$, $i = 1, 2, ...10$.

Цель суммирования с раундовыми константами – разрушить любую симметрию, что может возникнуть на разных этапах разворачивания ключа и привести к появлению слабых ключей, как в алгоритме DES.

Работа алгоритма расширения ключа продемонстрирована на рисунке.



ПРИМЕР. Начало зашифрования

Ключ шифра OF **15 71 C9 47 D9 E8 59 0C B7 AD DF AF 7F 67 98**

Раундовые ключи	Функция $g(w)$
w_0 =0F 15 71 C9 w_1 =47 D9 E8 59 w_2 =0C B7 AD DF w_3 =AF 7F 67 98	RotWord(w_3) = 7F 67 98 AF = x_1 SubWord(x_1) = D2 85 46 79 = y_1 R_{con} [1] = 01 00 00 00 y_1 + R_{con} [1] = D3 85 46 79 = z_1
w_4 = w_4 + z_1 = DC 90 37 B0 w_5 = w_4 + w_1 = 9B 49 DF E9 w_6 = w_5 + w_2 = 97 FE 72 3F w_7 = w_6 + w_3 = 38 81 15 A7	RotWord(w_7) = 81 15 A7 38 = x_2 SubWord(x_4) = 0C 59 5C 07 = y_2 R_{con} [2] = 02 00 00 00 y_2 + R_{con} [2] = 0E 59 5C 07 = z_2
w_8 = w_4 + z_2 = D2 C9 6B B7 w_9 = w_8 + w_5 = 49 80 B4 5E w_{10} = w_9 + w_6 = DE 7E C6 61 w_{11} = w_{10} + w_7 = E6 FF D3 C6	RotWord(w_{11}) = FF D3 C6 E6 = x_3 SubWord(x_2) = 16 66 B4 8E = y_3 R_{con} [3] = 04 00 00 00 y_3 + R_{con} [3] = 12 66 B4 8E = z_3

§21. РАСШИФРОВАНИЕ AES

Для расшифрования шифротекста все используемые шифрующие преобразования могут быть инвертированы и применены в обратном порядке. Перед первым раундом дешифрования выполняется операция AddRoundKey , накладывающая на шифротекст четыре последних слова расширенного ключа. Затем выполняется 10 раундов дешифрования, каждый из которых осуществляет такие операции:

1. Операция InvShiftRows, обратная операции ShiftRows. Байты в последних трех строках матрицы *State* циклически сдвигаются влево на различное число байт. Первая строка неподвижна, а нижние три строки сдвигаются влево на 1, 2 и 3 байта соответственно.

2. Операция InvSubBytes, обратная операции SubBytes. Байты матрицы *State* заменяются новыми значениями по приведенной ниже таблице обратной замены, являющейся инвертированным *S*-боксом.

	Таблица преобразования InvSubBytes															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DE	6E
A	47	E1	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	D	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7E	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	CB	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

3. Операция InvMixColumns – процедура, обратная процедуре MixColumns. Каждый столбец матрицы *State* рассматривается как четырехчленный многочлен над полем $GF(2^8)$ и умножается на фиксированный многочлен

$$c^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$$

по модулю многочлена $x^4 + 1$. Такую операцию можно записать в матричном виде

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \cdot \begin{pmatrix} s_0' \\ s_1' \\ s_2' \\ s_3' \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}, \quad \text{где} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \text{– столбец матрицы } State.$$

4. Операция AddRoundKey является обратной сама к себе, так как состоит только в суммировании по $\mathrm{mod}\,2$.

Последний раунд расшифрования не содержит операцию InvMixColumns.

В алгоритме расшифрования последовательность преобразований отличается от порядка операций шифрования, а алгоритм расширения ключа остается неизменным. Однако два свойства алгоритма AES позволяют построить другую эквивалентную процедуру расшифрования, где последовательность операций преобразования остается той же самой (естественно с заменой операций на обратные). Это

- коммутативность операций SubBytes и ShiftRows;
- линейность операций перемешивания в столбце MixColumns и InvMixColumns по отношению к данным столбца, что означает

$$\mathrm{InvMixColumns}(State \oplus \mathrm{RoundKey}) = \mathrm{InvMixColumns}(State) \oplus \mathrm{InvMixColumns}(\mathrm{RoundKey}).$$

Это дает возможность инвертировать порядок выполнения процедур InvSubBytes и InvShiftRows. Если же дополнительно в последовательности раундовых ключей изменить слова с помощью процедуры InvMixColumns (не трогая первые и последние четыре слова), то порядок выполнения процедур AddRoundKey и InvMixColumns тоже можно изменить на обратный.

§22. БЕЗОПАСНОСТЬ AES

Для трех вариантов ключей AES полный перебор требует 2^{127}, 2^{191} или 2^{255} операций соответственно. Даже наименьшее из этих чисел свидетельствует, что атака с использованием перебора ключей сегодня не имеет практического значения.

В соответствии с оценками разработчиков шифр устойчив против таких видов криптоаналитических атак:

- дифференциального криптоанализ;
- линейного криптоанализ;
- криптоанализ на основе связанных ключей (слабых ключей в алгоритме нет).

В июне 2003 года АНБ США объявило, что шифр AES является достаточно надёжным для защиты сведений, составляющих государственную тайну. Вплоть до уровня SECRET было разрешено использовать ключи длиной 128 бит, для уровня TOP SECRET рекомендовано ключи длиной 192 и 256 бит.

После введения нового стандарта шифрования AES попытки его вскрытия значительно увеличились. Комбинация метода бумеранга и связанных ключей вывела Алекса Бирюкова и Дмитрия Ховратовича на вскрытие в июле 2009 р. версий AES-192 и AES-256 (все раунды). Обе атаки проведены в предположении, что криптоаналитик перехватил пары «открытый текст – шифротекст», полученные на разных секретных ключах. Хотя атака срабатывает против любого ключа алгоритма, она сейчас, и вероятно останется навсегда теоретической, так как вычислительная сложность атаки 2^{119} – за пределами наших компьютеров. Интересно, что авторы отмечают «небоеспособность» нападения против версии AES-128, хотя, по иронии, именно ключ 256 битов предназначался для шифрования самой секретной информации, нежели ключ в 128 битов.

На конференции Crypto-2011 исследователи Андрей Богданов, Дмитрий Ховратович и Кристиан Речбергер представили новый метод криптоанализа – метод биклик, с помощью которого они восстановили ключ AES-128 с вычислительной сложностью $2^{126,1}$, а ключ AES-256 – со сложность $2^{254,1}$. Это разновидность атаки с подобранным шифротекстом. Вместо перебора ключей предлагается посчитать набор биклик (зависимостей шифротекстов от внутренних состояний для фрагментов разных ключей). Упрощённо – это перебор ключа по частям. Однако, усилия на нее придется затратить гигантские: количество итераций для поиска ключа к AES-128 выражается числом $8 \cdot 10^{37}$; у 1 трлн компьютеров, способных проверять по 1 млрд ключей в секунду, на поиск ключа уйдет 2 млрд лет. Для сравнения, ныне существующие компьютеры могут проверять лишь по 10 млн ключей за секунду.

Эту ситуацию известный криптограф Брюс Шнайер комментирует так: «Я не думаю, что сегодня существует какая-либо практическая атака на AES. Если Вы знаете, как взломать n-раундовый шифр, то удвойте или утройте число раундов. Это может быть или AES-128 с 16, или AES-192 с 20, или AES-256 с 28 раундами. А полного пересмотра стандарта снова и снова мы не хотим» (2009).

Единственный работающий способ взлома шифра AES – это атаки по побочным каналам. Такие атаки не связаны с математическими особенностями AES, а используют определённые особенности реализации систем, использующих шифр, с целью раскрыть частично или полностью секретные данные, в том числе ключ. Так, в апреле 2005

года Daniel J. Bernstein опубликовал работу с описанием атаки на основе информации о времени выполнения каждой операции шифрования. Данная атака потребовала более 200 миллионов выбранных шифротекстов для нахождения ключа. Другая атака Даг Арне Освика, Ади Шамира и Эран Трумера, также основанная на временном анализе выполнения операций, в октябре 2005 года уже раскрывала ключ всего лишь за 800 операций шифрования. Но от атакующий в этом случае должен запускать программы на той же системе, где выполнялось шифрование. В декабре 2009 года опубликована работа, в которой использование дифференциального анализа ошибок позволило восстановить ключ за 2^{32}операций (это разновидность криптоанализа на основе создания случайных аппаратных ошибок).

Осенью 2012 г. Сергей Скоробогатов, выпускник МИФИ, сотрудник группы безопасности компьютерной Лаборатории Кембриджского университета, опубликовал результаты тестирования нового метода чипов. Исследование было проведено в связи с заявлениями представителей разведывательных агентств, в том числе MI5 и АНБ, что микросхемы могут содержать бэкдоры, помещённые туда китайскими производителями. Кроме того, Великобритания опасается хакерских атак со стороны Китая, нацеленных на бизнес, военный сектор. В качестве примера подобной атаки приводится пример червя Stuxnet, нанесший в 2010 году урон ядерной промышленности Ирана.

Для изучения выбрали чип ProASCI3 FPGA китайского производства. После сканирования чипа на наличие необычных функций был обнаружен бэкдор, помещённый туда производителем и способный снять криптографическую защиту с микросхемы, поменять ключ AES, получить доступ к незашифрованным данным или вывести устройство из строя. Метод Скоробогатова – разновидность традиционного криптоанализа по потребленной мощности – получил название «трубопроводный анализ излучения».

§23. ВОПРОСЫ РЕАЛИЗАЦИИ И ПРИМЕНЕНИЯ AES

Алгоритм обладает не только очень высокой защищенностью, но и очень высокой скоростью шифрования. Программная реализация на машине с частотой 2 ГГц позволяет шифровать данные со скоростью 700 Мбит/с. Такой скорости достаточно для шифрации видео в формате MPEG-2 в реальном масштабе времени. Аппаратные реализации работают еще быстрее. В последнее время появилась новая версия AES-NI (New Instructions), которая позволяет оптимизировать работу алгоритма (понизить загрузку процессора на 50%). Эта версия может использоваться и совместно с SSL (SSL – криптографический протокол, который обеспечивает установление безопасного соединения между клиентом и сервером). Компания Intel разработала микросхему,

реализующую этот алгоритм (серия X5600). Количество клиентов при работе с версией AES-NI увеличивается на 13%.

ДОПОЛНЕНИЕ: БИБЛИОГРАФИЧЕСКИЕ СВЕДЕНИЯ

Джоан Дамен (англ. Joan Daemen, родился в 1965г.) – бельгийский криптограф.

После окончания Католического университета Левена (Бельгия) по специальностям «Электромеханика» и «Гражданское строительство» работал в исследовательской группе по компьютерной безопасности и промышленной криптографии (коллега В. Реймена). Занимаясь криптоанализом и проектированием новых шифров и криптографических хеш-функций, защитил докторскую диссертацию (1995 г.). Сегодня берет активное участие а разработке криптографических протоколов для смарт-карт, управления системами персонализации. Плодотворное сотрудничество с Рейменом завершилось появлением алгоритма RIJNDAEL. Другие работы – шифры MMB, SHARK, NOEKEON, 3-WAY, BASEKING. Недавно предложил криптографическую хеш-функцею KECCAK – одну из тех, что претендует на стандарт SHA-3.

Винсент Реймен (англ. Vincent Rijmen, родился в 1970 г.) – бельгийский криптограф.

Окончил Католический университет Левена (1993 г., Бельгия), после чего там же поступил в аспирантуру. В 1997 г. защитил докторскую диссертацию на тему «Криптоанализ и дизайн итеративних блокових шифров». Спельний проект с Даменом привел к алгоритму RIJNDAEL, что стал победителем конкурса на новый стандарт AES (2002 г.). Кроме того, Реймен – один из разработчиков криптографической хеш-функции WHIRLPOOL, блокових шифров ANUBIS KHAZAD, NOEKEON, SHARK. В 2001 – 2007 гг. – профессор Института прикладной обработки информации и коммуникаций Технологичного университета Граца (Австрия).

Брюс Шнайєр (англ. Bruce Schneier), (родился. 1963 г.) – известный американский криптограф.

Изучал информатику в Вашингтонском университете. Автор нескольких монографий, среди которых широко известная «Прикладная криптография: протоколы, алгоритмы и исходные тексты». Разработчик алгоритмов шифрования BLOWFISH, TWOFISH, SOLITAIRE, генератора псевдослучайных чисел YARROW. Член совета Международной ассоциаци криптографов IACR, президент консалтинговой компании Counterpane Systems в области криптографии и защиты информации, которая с 1991 до 1999 года оказывала услуги по проектированию шифров и их анализу крупнейшим компьютерным и финансовым корпорациям мира. Последнее время сотрудничает с правительственными спецслужбами в сфере обеспечения информационной безопасности. Выпускает свободно доступный ежемесячный Інтернет-информационный бюллетень по проблемам криптологии.