

Московский физико-технический институт
(государственный университет)

Курс семинаров по предмету "Защита информации"
Эссе

Алгоритм Rijndael

Глаз Роман Сергеевич
Группа Б01-008а

Долгопрудный
2023

Содержание

1	Введение	1
2	Принцип работы	1
2.1	Краткое описание	1
2.2	Описание процедуры трансформации round	2
2.2.1	Про раундовые ключи	2
2.2.2	Описание процедуры	2
2.3	Описание вспомогательных процедур	2
2.3.1	Процедура <i>AddRoundKey</i>	2
2.3.2	Процедура <i>SubBytes</i>	3
2.3.3	Процедура <i>ShiftRows</i>	3
2.3.4	Процедура <i>MixColumns</i>	4
2.3.5	Алгоритм генерации раундовых ключей <i>KeyExpansion</i>	4
3	Список используемой литературы	4

1. Введение

Rijndael на данный момент является стандартом шифрования правительства США по результатам проведённого конкурса *Advanced Encryption Standard*, организованного Национальным институтом стандартов и технологий США.

Потребности принятия нового стандарта возникли из-за того, что предыдущий стандарт – *Data Encryption Standard* – имел ключ длиной всего в 56 бит, что позволяло взломать шифр простым перебором ключей.

Алгоритм *Rijndael* стал настолько популярным, что даже производители процессоров Intel и AMD ввели аппаратную поддержку инструкций, ускоряющих работу *Rijndael*.

2. Принцип работы

2.1. Краткое описание

Пусть имеется набор входных данных I и ключ K , а B – количество 32-битных слов, из которых состоят ключ и входные данные, то есть $I = (i_1, \dots, i_B, \dots, i_{4B})$ и $K = (k_1, \dots, k_V, \dots, k_{4V})$. Возможные значения B : 4, 5, 6, 7 и 8. Возможные значения V : 4, 5, 6, 7 и 8.

Rijndael сводится к следующей формальной процедуре: получить согласно некоторым правилам шифро-текст $C = (c_1, \dots, c_B, \dots, c_{4B})$.

Введём понятие S (state) – текущее состояние алгоритма, которое в начале соответствует входным данным I , в процессе применения алгоритма соответствует некоторому промежуточному представлению, а после применения алгоритма – шифро-тексту C . S является матрицей размером 4 x B .

Алгоритм состоит из следующих процедур:

1. Исходные данные помещаются в текущее состояние S по следующему правилу:

$$S = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1B} \\ \dots & & & \\ s_{41} & s_{42} & \dots & s_{4B} \end{bmatrix} = \begin{bmatrix} i_1 & i_2 & \dots & i_B \\ \dots & & & \\ i_{3B+1} & i_{3B+2} & \dots & i_{4B} \end{bmatrix} \quad (1)$$

2. К состоянию S применяется процедура трансформации – раунд (round) – $N_R - 1$ раз, где N_R может принимать значения от 10 до 14 включительно в зависимости от длины ключа K (10 раз соответствует минимальной длине ключа 128 бит и т.д.). Полное описание раунда изложено в главе 2.2.
3. К состоянию S применяется последний раунд N_R – он немного отличается от предыдущих (подробнее об этом позже, см главу 2.2.2).
4. Состояние S благополучно копируется в шифро-текст C :

$$C : \begin{bmatrix} c_1 & c_2 & \dots & c_B \\ \dots & & & \\ c_{3B+1} & c_{3B+2} & \dots & c_{4B} \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1B} \\ \dots & & & \\ s_{41} & s_{42} & \dots & s_{4B} \end{bmatrix} \quad (2)$$

2.2. Описание процедуры трансформации round

2.2.1. Про раундовые ключи

Для каждого раунда генерируется собственный раундовый ключ W_r , представляющий собой матрицу такого же размера, что и матрица текущего состояния S , причём r – номер раунда. Имеется массив раундовых ключей размером $(N_R + 1)$: $W = (W_0, \dots, W_{N_R})$ (будем считать нумерацию раундов с нуля).

Процедура генерации раундовых ключей W_r называется "Расширение ключа" (*KeyExpansion*, подробнее в главе 2.3.5), в процедуре используется поданный на вход шифро-ключ K .

2.2.2. Описание процедуры

Процедура round при $0 \leq r < N_R$ над текущим состоянием S состоит из следующих этапов:

1. Применение процедуры "Сложить S с ключом раунда W_r " (*AddRoundKey*, подробнее в главе 2.3.1).
2. Применение процедуры "Использовать нелинейную таблицу замен для S " (*SubBytes*, подробнее в главе 2.3.2).
3. Применение процедуры "Сдвинуть строки в S " (*ShiftRows*, подробнее в главе 2.3.3).
4. Применение процедуры "Перемножить колонки S с полиномом" (*MixColumns*, подробнее в главе 2.3.4).

Процедура round при $r = N_R$, как уже было сказано, слегка отличается от предыдущих:

1. Применение процедуры "Сложить S с ключом раунда W_{N_R} " (*AddRoundKey*, подробнее в главе 2.3.1).
2. Применение процедуры "Использовать нелинейную таблицу замен для S " (*SubBytes*, подробнее в главе 2.3.2).
3. Применение процедуры "Сдвинуть строки в S " (*ShiftRows*, подробнее в главе 2.3.3).
4. Применение процедуры "Сложить S с ключом раунда" (*AddRoundKey*, подробнее в главе 2.3.1).

2.3. Описание вспомогательных процедур

2.3.1. Процедура *AddRoundKey*

Процедура может быть описана следующим образом: имеется текущее состояние S , описываемое в виде матрицы, и раундовый ключ W_r , представляющий собой матрицу такого же размера, что и S : новое состояние получается операциями

$$S := \begin{pmatrix} s_{11} \oplus w_{11} & s_{12} \oplus w_{12} & \dots & s_{1B} \oplus w_{1B} \\ \dots & \dots & \dots & \dots \\ s_{41} \oplus w_{41} & s_{42} \oplus w_{42} & \dots & s_{4B} \oplus w_{4B} \end{pmatrix} \quad (3)$$

2.3.2. Процедура *SubBytes*

Для процедуры *SubBytes* требуется таблица замен S-box: именно благодаря этой операции обеспечивается нелинейность алгоритма шифрования. Рассмотрим подробнее алгоритм генерации таблицы замен S-box.

Для начала определим $x^8 + x^4 + x^3 + x + 1$ – неприводимый многочлен в поле Галуа $GF(2^8)$. В дальнейшем будем работать только в этом поле.

S-box – таблица размерами 16 x 16 байт, в которой изначально ij -ый байт имеет значение $16 \cdot (i - 1) + j - 1$. Каждый байт b матрицы S-box заменяется обратным ему элементом b^{-1} . Заметим, что элементом со значением 0 не имеет обратного – присвоим ему значение 63_{16} .

Далее, выберем произвольный байт $Sbox_{ij} = c$, который преобразуем следующим образом:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{pmatrix} := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \quad (4)$$

где c_i – i -ый бит байта c .

Более коротким способом эту формулу можно записать как

$$c_i := c_i \oplus c_{(i+4) \bmod 8} \oplus c_{(i+5) \bmod 8} \oplus c_{(i+6) \bmod 8} \oplus c_{(i+7) \bmod 8} \oplus d_i, \quad (5)$$

где d_i – i -ый бит числа $63_{16} = 01100011_2$.

Продельвая такие же шаги для всех остальных байтов $Sbox_{ij}$ матрицы S-box, получим готовую таблицу замен.

Остаётся лишь ей воспользоваться: в матрицей текущего состояния S берём байт $s_{ij} = (x1 \ x2 \ x3 \ x4 \ y1 \ y2 \ y3 \ y4)$, где x_i, y_i – последовательные биты байта s_{ij} . Тогда пусть $x1 \ x2 \ x3 \ x4_2 + 1$ – номер строки в таблице замен, $y1 \ y2 \ y3 \ y4_2 + 1$ – номер столбца в таблице замен, сопоставляем по этим номерам новое значение s_{ij} , полученное с помощью S-box.

2.3.3. Процедура *ShiftRows*

Процедура может быть описана следующим образом: имеется текущее состояние S , а новое состояние получается путем циклического сдвига влево строк матрицы S : i -ая строка сдвигается на $(i - 1)$ байт.

В матричном виде это может быть записано как

$$S := \begin{pmatrix} s_{11} & s_{12} & \dots & s_{1(B-1)} & s_{1B} \\ s_{22} & s_{23} & \dots & s_{2B} & s_{21} \\ s_{33} & s_{34} & \dots & s_{31} & s_{32} \\ s_{44} & s_{45} & \dots & s_{42} & s_{43} \end{pmatrix} \quad (6)$$

Однако у алгоритма Rijndael есть одна особенность: если $B = 8$, тогда смещения нужно делать не на 0, 1, 2 и 3 байта соответственно, а на 0, 1, 3 и 4 байта.

2.3.4. Процедура *MixColumns*

Данная процедура считает каждый столбец матрицы состояния S как коэффициенты полинома третьей степени в поле Галуа $GF(2^8)$. Каждый столбец умножается на полином $3x^3 + x^2 + x + 2$ по модулю $x^4 + 1$. Данная процедура нужна для того, чтобы внести дополнительную диффузию в текущее состояние S .

Формально процедура может быть описана следующими формулами:

$$(s'_{1j} \cdot x^3 + s'_{2j} \cdot x^2 + s'_{3j} \cdot x^1 + s'_{4j}) = \quad (7)$$

$$= ((s_{1j} \cdot x^3 + s_{2j} \cdot x^2 + s_{3j} \cdot x^1 + s_{4j}) \cdot (3x^3 + x^2 + x + 2)) \pmod{x^4 + 1}, \quad (8)$$

$$s_{1j} := s'_{1j}, \dots, s_{4j} := s'_{4j}. \quad (9)$$

2.3.5. Алгоритм генерации раундовых ключей *KeyExpansion*

Рассмотрим подробнее генерацию раундовых ключей W_r , представляющих собой матрицы того же размера, что и матрица состояния S .

С самого начала имеется шифро-ключ $K = (k_1, \dots, k_V, \dots, k_{4V})$, состоящий из V 32-битных слов. Тогда алгоритм заполнения раундовых ключей выглядит следующим образом:

1. Для начала определим массив 32-битных слов $w[B \cdot (N_R + 1)]$. Пусть $i = 1$.
2. Пока $i \leq V$, выполняем: $w[i] = (k_i, k_{i+1}, k_{i+2}, k_{i+3})$, $i = i + 1$.
3. Пока $i \leq B \cdot (N_R + 1)$, выполняем:
 - (a) Пусть $a = w[i - 1]$.
 - i. Если $i = 0 \pmod{V}$, тогда $a = \text{SubWord}(\text{RotWord}(a)) \oplus \text{Rcon}[i/V]$.
 - ii. Иначе если $V > 6$ и $i = 4 \pmod{V}$, тогда $a = \text{SubWord}(a)$.
 - (b) Выполняем: $w[i] = w[i - V]$.
 - (c) Выполняем: $i = i + 1$.
4. Копируем массив $w[B \cdot (N_R + 1)]$ в каждый раундовый ключ по следующему правилу: последовательно заполняем столбцы матриц W_r , по очереди перекладывая 32-битные слова $w[i]$, последовательно по каждому раундовому ключу W_r .
5. В итоге имеем заполненные раундовые ключи W_r .

3. Список используемой литературы

- TBD
- TBD