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 *AddRoundKey* 2

2.3.2 Процедура *SubBytes* 3

2.3.3 Процедура *ShiftRows* 3

2.3.4 Процедура *MixColumns* 3

2.3.5 Алгоритм генерации раундовых ключей *KeyExpansion* 4

3 Список используемой литературы **4**

1. Введение

Rijndael на данный момент является стандартом шифрования привительства США по результатам проведённого конкурса *Advanced Encryption Standard*, огранизованного 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 $I = (i_1, \ldots, i_B, \ldots, i_{4B})$ и $K = (k_1, \ldots, k_V, \ldots, k_{4V})$. 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 $C = (c_1, \ldots, c_B, \ldots, c_{4B})$.

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:

$$S = \begin{Vmatrix} s_{11} & s_{12} & \ldots & s_{1B} \\ \ldots & & & \\ s_{41} & s_{42} & \ldots & s_{4B} \end{Vmatrix} = \begin{Vmatrix} i_1 & i_2 & \ldots & i_B \\ \ldots & & & \\ i_{3B+1} & i_{3B+2} & \ldots & i_{4B} \end{Vmatrix} \tag{1}$$

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:

$$C : \begin{Vmatrix} c_1 & c_2 & \ldots & c_B \\ \ldots & & & \\ c_{3B+1} & c_{3B+2} & \ldots & c_{4B} \end{Vmatrix} = \begin{Vmatrix} s_{11} & s_{12} & \ldots & s_{1B} \\ \ldots & & & \\ s_{41} & s_{42} & \ldots & s_{4B} \end{Vmatrix} \tag{2}$$

2.2. Описание процедуры трансформации round

2.2.1. Про раундовые ключи

Для каждого раунда генерируется собственный раундовый ключ W_r размером B 32-битных слов, где r – номер раунда. Вместе с исходным шифро-ключом $W_0 = K$ имеем массив ключей размером $B \cdot (N_R + 1)$ 32-битных слов: $W = (W_0, \dots, W_{N_R})$.

Процедура генерации раундовых ключей W_r называется "Расширение ключа" (*KeyExpansion, подробнее в главе 2.3.5*).

2.2.2. Описание процедуры

Процедура round при $0 \le r < N_R$ 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 $W_r = (w_1, \dots, w_B, \dots, w_{4B})$, новое состояние получается операцией

$$S := \begin{Vmatrix} s_{11} \oplus w_1 & s_{12} \oplus w_2 & \dots & s_{1B} \oplus w_B \\ \dots & & & \\ s_{41} \oplus w_{3B+1} & s_{42} \oplus w_{3B+2} & \dots & s_{4B} \oplus w_{4B}. \end{Vmatrix} \tag{3}$$

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}.

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

$$\begin{Vmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{Vmatrix} := \begin{Vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{Vmatrix} \cdot \begin{Vmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{Vmatrix} + \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix}, \tag{4}$$

где c_i - i-ый бит байта c.

Более коротким способом эту формулу можно записать как

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

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

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

Остаётся лишь ей воспользоваться: в матрицей текущего состояния S берём байт $s_{ij} = (\overline{x1\ x2\ x3\ x4}\ \ \overline{y1\ y2\ y3\ y4})$, где x_i, y_i – последовательные биты байта s_{ij}. Тогда пусть $\overline{x1\ x2\ x3\ x4}_2 + 1$ – номер строки в таблице замен, $\overline{y1\ y2\ y3\ y4} + 1$ – 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 матричмном виде это может быть записано как

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

2.3.4. Процедура *MixColumns*

TBD

2.3.5. Алгоритм генерации раундовых ключей *KeyExpansion*

TBD

3. Список используемой литературы

- TBD
- TBD