Rijndael

$Rijndael$[1] – итеративный блочный шифр. Длины блока шифруемых данных и секретного ключа могут принимать независимо друг от друга значения 128, 192 или 256 битов.

Блок $B = (b_0, b_1, \dots, b_{m-1})$ открытых данных, состоящих из $m = 16, 24$ или 32 байтов, в процессе криптографического преобразования представляется в виде матрицы $State$ (называемой *состоянием*), имеющей 4 строки и $NB = m/4$ столбцов. Аналогично, секретный ключ $K = (k_0, k_1, \dots, k_{q-1})$, состоящий из $q = 16, 24$ или 32 байтов, представляется в виде матрицы, имеющей 4 строки и $NK = q/4$ столбцов. Например, если $m = 21$ и $q = 16$, то B и K представляются в виде матриц:

$$B = \begin{pmatrix} \beta_{00} & \beta_{01} & \beta_{02} & \beta_{03} & \beta_{04} & \beta_{05} \\ \beta_{10} & \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} & \beta_{15} \\ \beta_{20} & \beta_{21} & \beta_{22} & \beta_{23} & \beta_{24} & \beta_{25} \\ \beta_{30} & \beta_{31} & \beta_{32} & \beta_{33} & \beta_{34} & \beta_{35} \end{pmatrix} = \begin{pmatrix} b_0 & b_4 & b_8 & b_{12} & b_{16} & b_{20} \\ b_1 & b_5 & b_9 & b_{13} & b_{17} & b_{21} \\ b_2 & b_6 & b_{10} & b_{14} & b_{18} & b_{22} \\ b_3 & b_7 & b_{11} & b_{15} & b_{19} & b_{23} \end{pmatrix},$$

$$K = \begin{pmatrix} \kappa_{00} & \kappa_{01} & \kappa_{02} & \kappa_{03} \\ \kappa_{10} & \kappa_{11} & \kappa_{12} & \kappa_{13} \\ \kappa_{20} & \kappa_{21} & \kappa_{22} & \kappa_{23} \\ \kappa_{30} & \kappa_{31} & \kappa_{32} & \kappa_{33} \end{pmatrix} = \begin{pmatrix} k_0 & k_4 & k_8 & k_{12} \\ k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \end{pmatrix}.$$

Другими словами, байты входного блока и ключа переписываются в соответствующие матрицы по столбцам. После завершения криптографического преобразования байты шифртекста получаются в том же порядке.

Число r раундов шифрования зависит от длины блока m и длины ключа q и определяется следующей таблицей:

$(m, q, r) =$

16, 16, 10	16, 24, 12	16, 32, 14
24, 16, 12	24, 24, 12	24, 32, 14
32, 16, 14	32, 24, 14	32, 32, 14

Например, если $m = 24$, $q = 16$, то $r = 12$.

На этапе предвычислений, предшествующему непосредственному шифрованию данных, секретный ключ K преобразуется в расширенный ключ – последовательность раундовых подключей K_0, K_1,…, K_r, каждый из которых имеет такой же размер, что и блок данных.

Один раунд криптографического преобразования, кроме заключительного, состоит из следующих операций над состоянием $State$ под управлением раундового подключа $RoundKey$:

$ByteSub(State)$;
$ShifRow(State)$;
$MixColumn(State)$;
$AddRoundKey(State, RoundKey)$.

Последовательность этих операций обозначается как $Round\ (State, RoundKey)$.

Заключительный раунд, обозначаемый как $FinalRound\ (State, RoundKey)$, отличается от $Round$ тем, что операция $MixColumn$ в нем отсутствует. (Это делает всю последовательность операций зашифрования симметричной.)

$ByteSub$ (замена байтов). Данное преобразование представляет собой замену байтов, выполняемую независимо над каждым байтом состояния. При этом байты интерпретируются как элементы конечного поля $\mathbb{F}_{256} \cong \mathbb{F}_2[x]/f(x)$, где $f(x) = x^8 + x^4 + x^3 + x + 1$ – неприводимый многочлен над $\mathbb{F}_2$ показателя 51. Операция замены является подстановкой $\pi = \rho \circ \tau$, $(\rho \circ \tau[x] \equiv \rho[\tau[x]])$, которая определяется как композиция следующих подстановок τ и ρ на множестве $\mathbb{F}_{256}$:

---

[1] Авторы шифра; *Joan Daemen* и *Vinsent Rijmen* (Бельгия)

1) $\tau: x \longrightarrow x^{254}$ (отметим, что $\pi(0) = 0$; если $x \neq 0$, то $\pi(x) = x^{-1}$);
2) подстановка ρ преобразует байт $x = x_7x_6x_5x_4x_3x_2x_1x_0$ в байт $y = y_7y_6y_5y_4y_3y_2y_1y_0$ (здесь x_i, y_i – биты в x и y) по правилу:
3)

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

Замечание. Значение $y = \rho[x]$ можно вычислить следующим образом:

$$y := x \oplus rol_4x \oplus rol_3x \oplus rol_2x \oplus rol_1x \oplus 0\text{x}63,$$

где rol_nx – циклический сдвиг байта x влево на n позиций, или

$$y := x \oplus shl_4x \oplus shl_3x \oplus shl_2x \oplus shl_1x \oplus shr_4x \oplus shr_5x \oplus shr_6x \oplus shr_7x \oplus 0\text{x}63,$$

где shl_nx (shr_nx) – сдвиг x влево (вправо) на n позиций.

Подстановки τ и ρ задают взаимо однозначные отображения на множестве $\mathbb{F}_2^8$, причем τ – нелинейное, а ρ – аффинное отображение. Подстановку π полезно задать в виде таблицы

$$(\pi[0], \pi[1], \ldots, \pi[255]).$$

Отметим также, что

$$\pi^{-1} = (\rho \circ \tau)^{-1} = \tau^{-1} \circ \rho^{-1} = \tau \circ \rho^{-1}.$$

Таблица подстановки π^{-1} вычисляется следующим образом:

$$\boldsymbol{for}\ \forall x \in \mathbb{F}_{256}\boldsymbol{do}\ \{y := \pi[x];\ \pi^{-1}[y] := x\}.$$

Операция, обратная к $ByteSub$, заключающегося в замене каждого байта x в $State$ на $\pi^{-1}[x]$, обозначается как $InvByteSub$.

$ShiftRow$ (сдвиг строк). Последние три строки $State$ циклически сдвигаются влево соответственно на c_1, c_2, и c_3 байта. Величины сдвигов зависят от длины блока:

$$(c_1, c_2, c_3) = \begin{cases} (1,2,3), & \text{если } m = 16 \text{ или } 24, \\ (1,3,4), & \text{если } m = 32. \end{cases}$$

Например, для 16-байтового блока это преобразование выглядит следующим образом:

$$\begin{pmatrix} \delta_{00} & \delta_{01} & \delta_{02} & \delta_{03} \\ \delta_{10} & \delta_{11} & \delta_{12} & \delta_{13} \\ \delta_{20} & \delta_{21} & \delta_{22} & \delta_{23} \\ \delta_{30} & \delta_{31} & \delta_{32} & \delta_{33} \end{pmatrix} \begin{matrix} \text{сдвига нет} \Rightarrow \\ \text{сдвиг на 1} \Rightarrow \\ \text{сдвиг на 2} \Rightarrow \\ \text{сдвиг на 3} \Rightarrow \end{matrix} \begin{pmatrix} \delta_{00} & \delta_{01} & \delta_{02} & \delta_{03} \\ \delta_{11} & \delta_{12} & \delta_{13} & \delta_{10} \\ \delta_{22} & \delta_{23} & \delta_{20} & \delta_{21} \\ \delta_{33} & \delta_{30} & \delta_{31} & \delta_{32} \end{pmatrix}$$

Операция, обратная к $ShiftRow$, возвращающая $State$ к исходному значению, обозначается как $InvShiftRow$.

Операции $ByteSub$ и *ShiftRow* перестановочны:

$$ByteSub \circ ShiftRow \equiv ShiftRow \circ ByteSub.$$

$MixColumn$ (перемешивание столбцов). Матрица $State$ умножается слева на невырожденную циркулянтную матрицу

$$A = \begin{pmatrix} 0\text{x}02 & 0\text{x}03 & 0\text{x}01 & 0\text{x}01 \\ 0\text{x}01 & 0\text{x}02 & 0\text{x}03 & 0\text{x}01 \\ 0\text{x}01 & 0\text{x}01 & 0\text{x}02 & 0\text{x}03 \\ 0\text{x}03 & 0\text{x}01 & 0\text{x}01 & 0\text{x}02 \end{pmatrix}$$

Как и в случае $Bytesub$, байты (элементы матрицы) интерпретируются как элементы того же поля $\mathbb{F}_{256}$. В результате $State$ получает новое значение:

$$State := \text{A} \times State;$$

Обратная операция $InvMixColumn$, возвращающая $State$ к исходному значению, реализуется путем умножения на матрицу A^{-1}, обратную к A:

$$InvMixColunm(State) \equiv \mathrm{A}^{-1} \times State,$$

где

$$\mathrm{A}^{-1} = \begin{pmatrix} 0\mathrm{x}0e & 0\mathrm{x}0\mathrm{b} & 0\mathrm{x}0\mathrm{d} & 0\mathrm{x}09 \\ 0\mathrm{x}09 & 0\mathrm{x}0e & 0\mathrm{x}0\mathrm{b} & 0\mathrm{x}0\mathrm{d} \\ 0\mathrm{x}0\mathrm{d} & 0\mathrm{x}09 & 0\mathrm{x}0e & 0\mathrm{x}0\mathrm{b} \\ 0\mathrm{x}0\mathrm{b} & 0\mathrm{x}0\mathrm{d} & 0\mathrm{x}09 & 0\mathrm{x}0e \end{pmatrix}.$$

AddRoundKey (добавление раундового подключа). К состоянию $State$ добавляется раундовый ключ $RoundKey$:

$$State := State \oplus RoundKey$$

посредством обычной побитовой операции $\oplus$ сложения по модулю 2. Операция $AddRoundKey$ инволютивна. Ее повторное применение возвращает $State$ к исходному значению.

Вычисление раундовых подключей

В алгоритме зашифрования используются $r + 1$ раундовых подключей, каждый из которых имеет такой же размер, что блок шифруемых данных. Для получения раундовых подключей на основе секретного ключа K строится *расширенный* ключ:

$$W = (W_0, W_1, \dots, W_p),$$

состоящий из $p = NB * (r + 1)$ 4-байтовых слов W_i. Раундовый ключ K_0 образует первые NB слов расширенного ключа, т.е.

$$K_0 = (W_0, W_1, \dots, W_{NB-1});$$

следующие NB слов образуют ключ K_1 и т.д. Так что:

$$K_i = (W_n, W_{n+1}, \dots, W_{n+NB-1}), n = i \cdot NB, i = 0,1 \dots$$

Первые NK слов W_i задаются секретным ключом K, т.е.:

$$(W_0, W_1, \dots W_{NK-1}) = K.$$

Остальные W_j ключей вычисляются рекурсивно, исходя из слов W_i с минимальными индексами:

for $i := NK$ **to** $Nb * (r + 1) - 1$ **do** {
    $temp := W_{i-1}$;
    if $(i \bmod NK) = 0$ **then** $temp := SubByte(RotByte(temp)) \oplus Rcon(i \operatorname{div} NK)$
    else if $(NK > 6)\&((i \bmod NK) = 4)$ **then** $temp := SubByte(temp)$;
    $W_i := W_{i-NK} \oplus temp$
}.

В этом описании $SubByte(W)$ обозначает замену каждого байта x в слове W на байт $\pi[x]$ (π – подстановка, используемая в $ByteSub$), а $RotByte\ (W)$ – сдвиг слова W влево на 24 позиции: слово (a, b, c, d) преобразуется в слово (b, c, d, a). Раундовые константы определяются как

$$Rcon[i] = (x^{i-1}, 0\mathrm{x}00, 0\mathrm{x}00, 0\mathrm{x}00), i = 1,2, \dots,$$

где $x = 0\mathrm{x}02$ – элемент поля $\mathbb{F}_{256}$.

Алгоритм зашифрования *Rijndael*

Вход: P – блок открытых данных (в виде $State$).
    $AddRoundKey\ (P, K_0)$;
    for $i := 1$ **to** $r - 1$ **do** $Round\ (P, K_i)$;
    $FinalRound\ (P, Kr)$.
Выход: P – блок шифртекста (в виде $State$).

При расшифровании обратные операции выполняются в обратном порядке:

Алгоритм расшифрования *Rijndael*

Вход: C – блок шифртекста (в виде $State$).

$AddRoundKey\ (C, K_r)$;
$InvShiftRow(C)$;
$InvByteSub(C)$;
$\boldsymbol{for}\ i := r-1\ \boldsymbol{downto}\ 1\ \boldsymbol{do}$ {
$AddRoundKey\ (C, K_i)$;
$InvMixColumn(C);\ InvShiftRow(C);\ InvByteSub(C)$
};
$AddRoundKey(C, K_0)$.

Выход: C – блок открытых данных (в виде $State$).

Замечание. Определим преобразования $InvRound$ и $InvFinalRound$ как

$InvRound(State, RoundKey) \equiv$ {
$InvBytSub(State)$;
$InvShiftRow(State)$;
$InvMixColumn(State)$
$AddRoundKey(State, RoundKey)$
};

$InvFinalRound(State, RoundKey) \equiv$ {
$InvByteSub(State)$;
$InvShiftRow(State)$;
$AddRoundKey(State, RoundKey)$
},

а последовательность раундовых подключей $K_0, \ldots, K_r$ преобразуем в $K_0^{-1}, \ldots, K_r^{-1}$, где

$K_0^{-1} := K_r$;
$\boldsymbol{for}\ i := 1\ \boldsymbol{to}\ r-1\ \boldsymbol{do}\ K_i^{-1} := A^{-1}K_{r-i}$;
$K_r^{-1} := K_0$.

Тогда алгоритм расшифрования преобразуется к виду:

$AddRoundKey(\mathrm{C}, K_0^{-1})$;
$\boldsymbol{for}\ i := 1\ \boldsymbol{to}\ r-1\ \boldsymbol{do}\ InvRound\ (\mathrm{C}, K_i^{-1})$;
$InvFinalRound(\mathrm{C}, K_r^{-1})$.