Cifrado simétrico en bloque

El algoritmo AES

eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

KISA

El algoritmo AES

En el año 2001, el NIST, National Institut for Standars and Technology, adoptaba el algoritmo *Rijndael*, desarrollado por los criptólogos Daemen y Rijmen, como nuevo *Estándar Avanzado de Cifrado (AES)* para su empleo en aplicaciones criptográficas no militares.

- Adoptado tras pasar un proceso de selección, revisión y estudio público y abierto.
- Fue diseñado para prevenir criptoanálisis diferencial y lineal.
- Emplea bloques y claves de longitud variable (128-256 bits).
- El número de rondas es función de las longitudes de bloque y clave.
- No posee estructura de red de Feistel.

Estructura del AES

El algoritmo AES se basa en aplicar un número determinado de *rondas* a un valor intermedio que se llama *estado*.

Cada ronda es una composición de cuatro funciones invertibles diferentes en tres niveles o *capas*.

- *Capa no lineal*:
  - Función **ByteSub**
- *Capa lineal*:
  - Función **ShiftRow**
  - Función **MixColumn**
- *Capa de adición de clave*:
  - Función **AddRoundKey**

En la última ronda no hay MixColumn

Estado

El estado puede representarse con una matriz de bytes de 4 filas y N_b columnas. N_b= número de bits del bloque dividido por 32.

Por ejemplo, para un bloque de 192 bits, $N_b = \frac{192}{32} = 6$.

$$\begin{array}{cccccc} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{array}$$

La matriz de estado puede considerarse como un vector de palabras. Cada columna es una palabra.

El bloque que se pretende cifrar o descifrar se traslada directamente byte a byte sobre la matriz de estado siguiendo la secuencia $a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, \ldots$

Clave

La clave tiene una estructura análoga a la del estado y se representa con una matriz de 4 filas y N_k columnas. N_k= número de bits de la clave dividido por 32.

Por ejemplo, para una clave de 128 bits, $N_k = \frac{128}{32} = 4$.

$$\begin{array}{cccc} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{array}$$

Los bytes de la clave se copian sobre la matriz de clave siguiendo la secuencia $k_{00}, k_{10}, k_{20}, k_{30}, k_{01}, \ldots$.

Número de rondas

El número de rondas n para AES depende de los tamaños de clave y bloque, según la tabla

n	$N_b = 4$ (128 bits)	$N_b = 6$ (192 bits)	$N_b = 8$ (256 bits)
$N_k = 4$ (128 bits)	10	12	14
$N_k = 6$ (192 bits)	12	12	14
$N_k = 8$ (256 bits)	14	14	14

Algoritmo AES con n rondas:

Si B es el bloque que queremos cifrar y S es la matriz de estado:

Paso 1. Calcular las subclaves $K_0, K_1, ..., K_n$ a partir de la clave original K.

Paso 2. Hacer $S = B \oplus K_0$.

Paso 3. Para $i = 1, \ldots, n$

Paso 3.1. Aplicar ronda i-ésima con la subclave K_i.

Rondas

Si S es la matriz de estado y K_i es la clave correspondiente a la ronda i-ésima, cada una de las rondas posee la siguiente estructura:

Paso 1. Hacer $S = ByteSub(S)$.

Paso 2. Hacer $S = ShiftRow(S)$.

Paso 3. Hacer $S = MixColumn(S)$.

Paso 4. Hacer $S = AddRoundKey(S, K_i)$.

En la última ronda se elimina el paso 3.

En el descifrado se utilizan las funciones inversas.

ByteSub

Sustitución no lineal de cada byte de la matriz de estado, mediante una S-Caja que se obtiene componiendo dos transformaciones:

1. Sustitución de cada byte por su inverso en $\mathrm{GF}(2^8)$:
$$a = a_7a_6a_5a_4a_3a_2a_1a_0 \longrightarrow x = a^{-1} = x_7x_6x_5x_4x_3x_2x_1x_0$$
   El valor cero queda inalterado.

2. Transformación afín:
$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

La función inversa de *ByteSub* será la aplicación de la inversa de la S-Caja correspondiente a cada byte de la matriz de estado.

ShiftRow

Desplazamiento cíclico a la izquierda en las filas de la matriz de estado. En cada fila f_i el desplazamiento es de un número de posiciones c_i diferente.
c_0 siempre es igual a cero (en esta fila no hay desplazamiento).
c_1, c_2, c_3 son función de N_b.

	c_0	c_1	c_2	c_3
$N_b = 4$	0	1	2	3
$N_b = 6$	0	1	2	3
$N_b = 8$	0	1	3	4

La función inversa de *ShiftRow* es un desplazamiento a la derecha de igual número de posiciones en las filas de la matriz de estado.

MixColumn

Producto (a nivel de palabra) de cada palabra del estado por $(03, 01, 01, 02)$. Es decir, producto módulo $M(x) = x^4 + 1$ de cada palabra por el polinomio

$$c(x) = 03x^3 + 01x^2 + 01x + 02,$$

donde los coeficientes están expresados en hexadecimal.

Recordemos que cada palabra se representa con un elemento de $\mathcal{P}_M(\mathrm{GF}(2^8))$: polinomio de grado menor que 4 cuyos coeficientes son elementos de $\mathrm{GF}(2^8)$.

Si $a_3a_2a_1a_0$ es una palabra (a_i byte),

$$a_3a_2a_1a_0 \leftrightarrow a(x) = a_3x^3 + a_2x^2 + a_1x + a_0.$$

La palabra resultante de la transformación es

$$b(x) \equiv c(x) \cdot a(x) \mod (x^4 + 1).$$

$$b(x) \equiv c(x) \cdot a(x) \quad \text{mód } (x^4+1).$$

Es decir,

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0,$$

donde

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

Observación: $c(x)$ posee inverso $\text{mód } (x^4+1)$ en $\mathcal{P}_M(\mathrm{GF}(2^8))$:

$$d(x) = c(x)^{-1} = 0bx^3 + 0dx^2 + 09x + 0e.$$

La función inversa de *MixColumn* es el producto (a nivel de palabra) de cada palabra por $(0b, 0d, 09, 0e)$.

InvMixColumn

$$d(x) = c(x)^{-1} = 0bx^3 + 0dx^2 + 09x + 0e.$$

$$a(x) \equiv d(x) \cdot b(x) \mod (x^4 + 1).$$

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

AddRoundKey

XOR de los bytes del estado con los bytes correspondientes de la subclave.

$$AddRoundKey(S, K_i) = S \oplus K_i.$$

$$\begin{array}{cccccc} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{array} \oplus \begin{array}{cccccc} k_{00} & k_{01} & k_{02} & k_{03} & k_{04} & k_{05} \\ k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\ k_{20} & k_{21} & k_{22} & k_{23} & k_{24} & k_{25} \\ k_{30} & k_{31} & k_{32} & k_{33} & k_{34} & k_{35} \end{array}$$

La función inversa de *AddRoundKey* es ella misma.

Descifrado

Para descifrar en AES hay que cambiar el orden de las subclaves y el de las funciones en cada ronda. Las funciones que se aplican son las inversas de las utilizadas en el cifrado.

Cifrado:

$$AK \mid BS \quad SR \quad MC \quad AK \mid \ldots \mid BS \quad SR \quad MC \quad AK \mid BS \quad SR \quad AK$$

Descifrado:

$$AK \quad ISR \quad IBS \mid AK \quad IMC \quad ISR \quad IBS \mid \ldots \mid AK \quad IMC \quad ISR \quad IBS \mid AK$$

En el cifrado y descifrado están intercambiados *ByteSub* con *ShiftRow* por una parte y *AddRoundKey* con *MixColumn* por otra.

$$\begin{array}{ccccccccccccc} AK & \underbrace{BS \quad SR} & \underbrace{MC \quad AK} & \ldots & \underbrace{BS \quad SR} & \underbrace{MC \quad AK} & \underbrace{BS \quad SR} & AK \\ AK & ISR \quad IBS & AK \quad IMC & \ldots & ISR \quad IBS & AK \quad IMC & ISR \quad IBS & AK \end{array}$$

- Como *ByteSub* opera en bytes mientras que *ShiftRow* sólo los cambia de lugar, las dos operaciones pueden intercambiarse.
- La secuencia

$$\begin{array}{l} AddRoundKey(S, RoundK) \\ InvMixColumn(S) \end{array}$$

  puede cambiarse por

$$\begin{array}{l} InvMixColumn(S) \\ AddRoundKey(S, InvRoundK) \end{array}$$

  donde *InvRoundK* se obtiene aplicando *InvMixColumn* a *RoundK*.

De esta forma, la estructura del descifrado es idéntica a la del cifrado, usando las funciones inversas.

$$\begin{array}{c|cccc|c|cccc|ccc} AK & BS & SR & MC & AK & \dots & BS & SR & MC & AK & BS & SR & AK \\ AK & IBS & ISR & IMC & AK^I & \dots & IBS & ISR & IMC & AK^I & IBS & ISR & AK \end{array}$$

Cálculo de las subclaves

Si n es el número de rondas, necesitamos $(n+1)$ subclaves $K_0, K_1, \ldots, K_n$.

Si la matriz de estado es $4 \times N_b$, entonces el número de bytes del bloque es $4N_b$ y necesitaremos en total $(n+1)4N_b$ bytes.

Las diferentes subclaves K_i se derivan de la clave principal K mediante el uso de dos funciones, una de expansión y otra de selección.

La función de expansión permite obtener, a partir de K, una secuencia de $4(n+1)N_b$ bytes ($(n+1)N_b$ palabras).

La función de selección simplemente toma consecutivamente de la secuencia obtenida bloques del mismo tamaño que la matriz de estado y los va asignando a cada K_i.

La función de expansión

La clave expandida es un vector de $N_b(n+1)$ palabras, denotado por W.

Las primeras N_k palabras contienen la clave original K.

Las demás palabras se definen recursivamente en términos de palabras anteriores.

Hay dos algoritmos, uno para $N_k \leq 6$ y otro para $N_k > 6$.

En los dos algoritmos se utilizan las funciones *SubByte* y *RotByte* y unas constantes denotadas $Rcon(j)$.

Función de expansión para $N_k > 6$

Paso 1. Para $i = 0, \ldots, N_k - 1$

  Paso 1.1. Hacer $W[i] = (K[4i], K[4i+1], K[4i+2], K[4i+3])$

Paso 2. Para $i = N_k, \ldots, N_b(n+1) - 1$

  Paso 2.1. Hacer $tmp = W[i-1]$

  Paso 2.2. Si $i \equiv 0 \mod N_k$

    Paso 2.2.1. Hacer

$$tmp = SubByte(RotByte(tmp)) \oplus Rcon(i/N_k)$$

  Paso 2.3. Si $i \equiv 4 \mod N_k$

    Paso 2.3.1. Hacer $tmp = SubByte(tmp)$

  Paso 2.4. Hacer $W[i] = W[i - N_k] \oplus tmp$

Función de expansión para $N_k \leq 6$

Es el mismo algoritmo suprimiendo el paso 2.3.

$$N_k = 4 \qquad \underbrace{0 \quad 1 \quad 2 \quad 3}_{K} \quad 4 = 0 + 3 \quad 5 = 1 + 4 \quad \ldots$$

La función *SubByte* devuelve el resultado de aplicar la S-Caja AES a cada uno de los bytes de la palabra.

La función *RotByte* es una permutación cíclica a la izquierda de los bytes de la palabra. $RotByte(a, b, c, d) = (b, c, d, a)$.

Las constantes $Rcon(j)$ se definen de la siguiente forma:

- $Rcon(j) = (RC(j), 00, 00, 00)$.
- $RC(j)$ es el elemento de $\mathrm{GF}(2^8)$ correspondiente a x^{j-1} módulo $m(x) = x^8 + x^4 + x^3 + x + 1$.

Por ejemplo,

$$RC(9) \equiv x^8 \equiv x^4 + x^3 + x + 1 \mod m(x).$$

Es decir, $RC(9) = 00011011\,(= 1b)$ y $Rcon(9) = (1b, 00, 00, 00)$.

Fin de la sección