Cátedra No. 6 : 21 de Agosto

Transcriptor: Rodrigo González

1 Descripción de DES

DES (Data Encryption Standard) encripta un texto plano x de una longitud de 64 bits, i.e. $x \in (\mathcal{Z}_2)^{64}$, usando una clave K la cual es un bitstring (un string en su representación en bits) de 56 bits, obteniendo un texto cifrado de 64 bits.

Veamos ahora un esquema del DES. El algoritmo consiste en 3 etapas:

1. Dado un texto plano x, un bitstring x_0 es constituído mediante permutación de los bits de x, de acuerdo a una permutacion inicial IP fija. Nosotros escribimos $x_0 = IP(x) = L_0R_0$, donde L_0 abarca los primeros 32 bits de x_0 y R_0 los últimos 32 bits.

2. 16 iteraciones de una función conocida es entonces calculada. Se calcula L_iR_i, $1 \leq i \leq 16$, de acuerdo a la siguiente regla:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i), \end{aligned}$$

   donde $\oplus$ denota el o-exclusivo de dos bitstrings. La función f se describirá mas adelante, y $K_1, K_2, \ldots, K_{16}$, son bitstrings de 48 bits, calculados como una función de las claves K. (De hecho, cada K_i es una selección permutada de bits de K.)

3. Aplicando la permutación inversa IP^{-1} a los bistring $R_{16}L_{16}$ se obteniene el texto cifrado y. Que es, $y = IP^{-1}(R_{16}L_{16})$. Notar el orden inverso de L_{16} y R_{16}.

Explicaremos ahora la función f. La función f toma como entrada un primer argumento A, el cual es un bitstring de longitud de 32 bits, y un segundo argumento J que es un bitstring de longitud de 48 bits, produciendo como salida un bitstring de longitud de 32 bits. Los siguientes pasos son ejecutados:

1. El primer argumeto A es "expandido" a un bitstring de longitud de 48 bits de acuerdo a una función de expanción E fija. $E(A)$ consiste en 32 bits de A, permutados de una cierta manera, con 16 de los bits apareciendo dos veces.

2. Se calcula $E(A) \oplus J$ y se escribe el resultado como la concatenación de 8 strings de 6-bits $B = B_1B_2B_3B_4B_5B_6B_7B_8$.

3. El próximo paso usa 8 cajas-S $S_1, \ldots, S_8$. Cada S_i es un arreglo de 4x16 fijo cuyas entradas se interpretan como enteros de 0 al 15. Dado un bitstring de largo 6, digamos $B_j = b_1b_2b_3b_4b_5b_6$, nosotros calculamos $S_j(B_j)$ como sigue; los dos bits b_1b_6 determinan la representación binaria de una fila r de S_j $(0 \leq r \leq 3)$, y los 4 bits $b_2b_3b_4b_5$ determinan la representación binaria de una columna c de S_j $(0 \leq c \leq 15)$. Entonces $S_j(B_j)$ está definido a ser la entrada $S_j(rc)$, escrita en binario como un bitstring de longitud 4. Asi, nosotros calculamos $C_j = S_j(B_j)$, $(0 \leq j \leq 8)$.

4. El bitstring $C = C_1C_2C_3C_4C_5C_6C_7C_8$ de longitud de 32 bits es permutado de acuerdo a una permutación P fija. El resultante bitstring $P(C)$ esta definido como $f(A, J)$.

2 Claves Intermedias

Veamos como generar $K_1, \ldots, K_{16} \in (\mathcal{Z}_2)^{48}$ a partir de $K \in (\mathcal{Z}_2)^{56}$

En la práctica K tiene 64 bits y 8 de ellos son bits de paridad (8,16,24,...,64) que se setean de forma que:

$k_i + \ldots + k_{i+7}$ sea impar, con $i = 1, 9, \ldots$, donde $K = (k_1, \ldots, k_{64})$

Para calcular las claves intermedias se procede de la siguiente manera:

- Dada una clave K de 64-bits, se descarta los bits de paridad y se permutan los restantes bits de K de acuerdo a una permutación $PC-1$ fija. Nosotros escribiremos $PC-1(K) = C_0D_0$, donde C_0 abarca los 28 primeros bits de $PC-1(K)$ y D_0 los últimos 28 bits.

- Para i recorriendo desde 1 a 16, calculamos:

$$\begin{aligned} C_i &= LS_i(C_{i-1}) \\ D_i &= LS_i(D_{i-1}), \end{aligned}$$

y $K_i = PC - 2(C_iD_i)$. LS_i es un shift cíclico a la izquierda de 1 o 2 bits, dependiendo del valor de i (shift de 1 bit si $i = 1, 2, 9, 16$ y shift de 2 bits en caso contrario). $PC - 2$ es otra permutación fija.

3 Controversias

Cuando el DES fue propuesto como estandar, fue muy criticado. Una de las objeciones al DES concierne a las cajas-S. Todos los calculos excepto los de las cajas-S son lineales. Las cajas-S, son las componentes no lineales del criptosistema, y son vitales para su seguridad. Mucha gente sugieren que las cajas-S pueden contener una "trapdoors" escondida, la cual le permitiria a la NSA decriptar el DES. Es imposible desmentir tal afirmación, ya que no hay evidencias que indiquen que esas "trapdoors" existan o no.

La principal crítica a DES es que el tamaño del espacio de claves, 2^{56}, es demasiado pequeño para que el criptosistema sea seguro. Varias maquinas especializadas han sido propuestas para un ataque de texto plano y de su correspondiente texto cifrado conocidos, las cuales esencialmente mejoran la búsqueda exhaustiva de la clave. Esto es, dado un texto plano x, de 64-bits, y su correspondiente texto cifrado y, cada clave posible será testeada hasta encontrar un clave K tal que $e_K(x) = y$ (notar que puede haber más de una).

4 Modo de Uso

A pesar que la descripción del DES es bastante larga, puede ser implimentado de manera muy eficiente, tanto en hardware como en sotfware. Las únicas operaciones aritméticas a ser realizadas son los o-exclusivos entre bitstrings. La función de expanción E, las cajas-S, las permutaciones IP y P, y los cálculos de $K_1, K_2, \ldots, K_{16}$ todos pueden ser hechos eficientemente mediante "table lookup" (Tablas permanentes que quedan en memoria para su acceso, en software) o mediante "hard-wiring" dentro de un circuito.

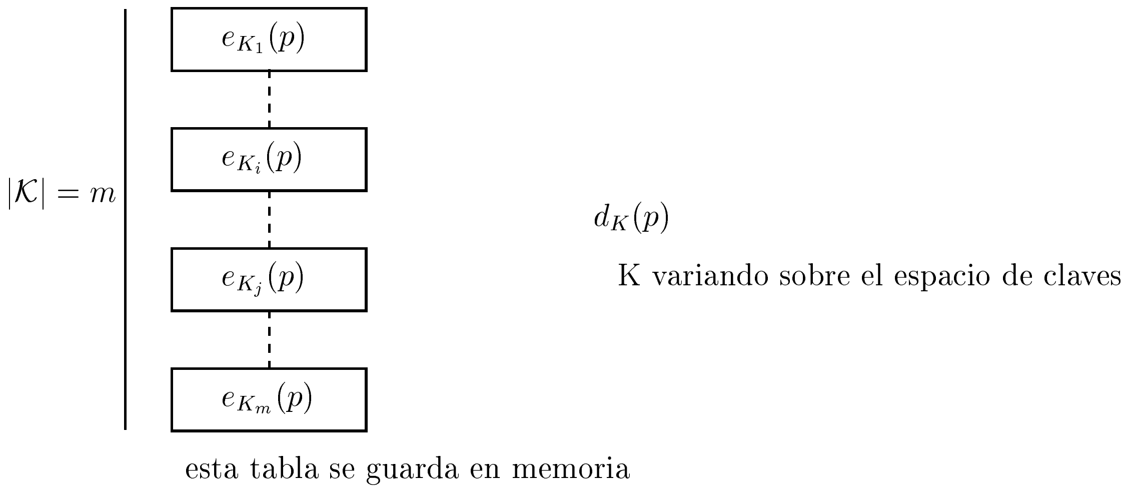

Figure 1: Tabla de busqueda de coincidencia de claves

4.1 DES Triple

La pregunta obvia es ¿por qué no iterar DES dos veces en vez de tres?.

En generar iterar dos veces un criptosistema los hace suceptibles a un ataque que se denomina "meet in the middle" [Merkle y Hellmam]. A continuación describiremos este ataque:

Sea $(\mathcal{P}, \mathcal{P}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ un criptosistema cualquiera. Para atacar S^2 se puede proceder de la siguiente forma:

1. Se elige un $p \in \mathcal{P}$ al azar.
2. Se determina $c = e_{K_2}(e_{K_1}(p))$ (el ataque es de texto plano conocido).
3. Se construye una tabla (ver figura 1)

La idea es que si calculamos $d_K(c)$ y se tiene que $e_{K_i}(p) = d_K(c)$ entonces K_i es un candidato a ser K_1, y K es un candidato a ser K_2.

Para verificar que los potenciales candidatos son los correctos, se verifica si $e_{K_i}(p_j) = d_K(c_j)\ \ j = 1, \ldots;$ donde $e_{K_2}(e_{K_1}(p_j)) = c_j$.

El ataque requiere $O(|\mathcal{K}|)$ memoria, esto ya que para todo K posible se calcula $e_K(p)$ y su resultado se guarda en memoria. Estudiemos el tiempo que demora el ataque, nos interesa que $N = \mathbb{E}_{p,k}[|\{k' : e_k(p) = e_{k'}(p)\}|]$ sea pequeño, (donde p, k indica sobre que parámetros se esta calculando la esperanza).

Para un buen criptosistema uno espera que:

$$N \simeq \frac{|\mathcal{K}|}{|\mathcal{P}|}.$$

Luego uno esperaría que:

$$\mathbb{P}r_{p,k}[e_k(p) = c] \simeq \frac{|\mathcal{K}|}{|\mathcal{K}||\mathcal{P}|} = \frac{1}{|\mathcal{P}|}.$$

Ya que, si tengo $p \in \mathcal{P}$, y aplico e_K sobre p obtengo que $e_K(p) \in \mathcal{C} = \mathcal{P}$, luego si se supone que dado un $c \in \mathcal{C}$, $|p : e_K(p) = c| = cte$, i.e. se sigue una distribución uniforme sobre las imagenes de e_K, obteniendo el resultado anterior. Por otro lado tenemos que:

$$\begin{aligned} N &= |\mathcal{K}|\mathbb{E}_{p,k}[\mathbb{P}r_{k'}[e_k(p) = e_{k'}(p)]] \\ &= |\mathcal{K}|\mathbb{P}r_{p,k,k'}[e_k(p) = e_{k'}(p)] \\ &= |\mathcal{K}|\textstyle\sum_{c\in\mathcal{P}} \mathbb{P}r_{p,k,k'}[e_k(p) = c, e_{k'}(p) = c] \\ &\simeq |\mathcal{K}|\textstyle\sum_{c\in\mathcal{P}} \mathbb{P}r_{p,k}[e_k(p) = c]^2 \simeq \frac{|\mathcal{K}|}{|\mathcal{P}|}. \end{aligned}$$

En el último paso se supone que $\mathbb{P}r_{p,k,k'}[e_k(p) = c, e_{k'}(p) = c] \simeq \mathbb{P}r_{p,k}[e_k(p) = c]^2$, ya que los eventos $e_k(p) = c$ y $e_{k'}(p) = c$ son más o menos independientes, esto se puede ver en que la elección de la clave K es totalmente aleatoria sobre el espacio de claves.

Luego el tiempo esperado que demora del ataque es $O(\frac{|\mathcal{K}||\mathcal{K}|}{|\mathcal{P}|})$, ya que $\frac{|\mathcal{K}|}{|\mathcal{P}|}$ representa $\mathbb{E}_{p,k}[|\{k' : e_k(p) = e_{k'}(p)\}|]$,es decir, el valor esperado de coincidencias con la clave; y que además esto se hace para cada $K \in \mathcal{K}$, es decir, $|\mathcal{K}|$ veces.

4.2 Otros modos de uso del DES

Otros 4 modos de operación han sido desarrollados para el DES.

ECB (electronic codebook mode)

El modo ECB corresponde al usual uso de un cifrado por bloques: dada una secuencia $x_1 x_2 \ldots$ de un texto plano por bloques de 64 bits, cada x_i es encriptado con la misma clave K, produciendo un string de bloques de texto cifrado, $y_1 y_2 \ldots$.

CBC (cipher block chaining mode)

En el modo CBC, cada bloque de texto cifrado y_i es x-ored (hacer el o-exclusivo con x) con el próximo bloque de texto plano x_{i+1} antes de ser encriptado con la clave K. Formalmente, se parte con un vector de inicialización de 64 bits IV, y definimos $y_0 = IV$. Entonces construimos $y_1, y_2, \ldots$ con la regla $y_i = e_K(y_{i-1}) \oplus x_i, i \geq 1$.

OFB (output feedback mode) y CFB (cipher feedback mode)

En los modos OFB y CFB, una keystream (flujo de claves) es generado, el cual es x-ored con el texto plano. OFB es un criptosistema de chorro, el chorro de claves es producido mediante la encripción reiterada de un vector de inicialización IV. Se define $z_0 = IV$, y entonces se calcula el chorro de claves $z_1 z_2 \ldots$ con la regla $z_i = e_K(z_{i-1}), i \geq 1$. La secuencia de texto plano $x_1 x_2 \ldots$ es entonces encriptada calculando $y_i = x_i \oplus z_i, i \geq 1$.

En el modo CFB , se parte con $y_0 = IV$ (vector de inicialización) y se produce un chorro de claves $z_1 z_2 z_3 \ldots$ mediante la encripción del anterior bloque cifrado. Esto es, $z_i = e_K(y_{i-1}), i \geq 1$. Como en el modo OFB, $y_i = x_i \oplus z_i, i \geq 1$.

MAC (message authentication code)

La propiedad de que los bloques cifrados dependan de todos los anteriores, es decir, que los y_i dependen de los x_j con $j \leq i$ es una propiedad que es útil para efectos de autentificación, esto ocurre en los modos CBC y CFB.

Describiremos como el modo CBC es usado para producir un MAC. Empezamos con un vector de inicialización IV consistente de puros ceros. Entonces se construyen los bloques de texto cifrado $y_1, \ldots, y_n$ con la clave K, usando CBC. Finalmente, definimos el MAC como y_n. Luego Alicia transmite la secuencia de texto plano , $x_1, \ldots, x_n$, junto con el MAC. Entonces B recibe $x_1, \ldots, x_n$, y puede reconstruir $y_1, \ldots, y_n$ usando la clave K (secreta), y verificar que y_n es el mismo MAC que se recibio.

La idea es que B puede verificar la integridad del mensaje recibido $x_1, \ldots, x_n$ calculando su MAC y verificar que corresponde al MAC transmitido.

Podemos combinar el proceso anterior con encripción de dos formas:

1. Tomar $x_1, \ldots, x_n$ generar el MAC de $x_1, \ldots, x_n$, digamos x_{n+1} y encriptar $x_1, \ldots, x_{n+1}$ y enviarlo.

2. Tomar $x_1, \ldots, x_n$ encriptar y obtener $y_1, \ldots, y_n$. Calcular el MAC de $y_1, \ldots, y_n$, digamos y_{n+1} y enviar $y_1, \ldots, y_{n+1}$