Hashing

Este documento no pretende ser un curso exhaustivo, en el mejor de los casos, únicamente puede considerarse como un conjunto de notas complementarias en alguna asignatura. Por supuesto, todo documento es susceptible de mejora, cualquier sugerencia o comunicación de error u omisión será bienvenida.

Una función resumen o función hash es una función h cuyo dominio está formado por mensajes de longitud arbitraria (habitualmente binarios) y cuya salida es una cadena (binaria) de longitud fija y prefijada. La secuencia de salida se reduce habitualmente a unos pocos bits, por lo que h no es inyectiva y se producen *colisiones*. Sin embargo, dada una función hash h aleatoria que admita mensajes binarios de tamaño t y proporcione secuencias binarias de tamaño n, de acuerdo con una distribución uniforme, a cada mensaje le corresponden $2^t/2^n$ mensajes, siendo la probabilidad de que dos mensajes colisionen de 2^{-n}.

Para un uso criptográfico, la posibilidad de encontrar colisiones eficientemente supone una debilidad, ya que el interés de estas funciones en criptografía se debe a la dificultad de encontrar estas colisiones.

Esquema Damgard-Merkle

Las funciones resumen para uso criptográfico deben garantizar determinadas propiedades. Un esquema ampliamente utilizado en el diseño de funciones hash computa el resumen considerando un *estado* que se modifica iterativamente al procesar el mensaje. Este esquema se conoce como de *Damgard-Merkle* y se resume en el Algoritmo 1.

La estrategia de procesamiento en cascada provoca que ligeras modificaciones de la entrada den como resultado resúmenes muy diferentes. Como ejemplo, consideramos la función resumen MD5 que sigue el esquema de Damgard-Merkle, el procesamiento del mensaje:

```
ESTO ES UN EJEMPLO DE MENSAJE PARA RESUMIR
```

da como resultado el resumen:

```
c38586997dfc8a053119f0cfc1d8d124
```

sin embargo, un pequeño cambio en la entrada:

```
ESTO FS UN EJEMPLO DE MENSAJE PARA RESUMIR
```

produce un resumen muy diferente:

```
269497fc078a0403f370bd26240c04cd
```

Algoritmo 1 Esquema de Damgard-Merkle

Entrada: Mensaje x (binario) de longitud arbitraria n
Salida: Resumen del mensaje $h(x)$ de longitud fija k

$//n >> k$

1: Dividir el mensaje en una serie de bloques de tamaño fijo t $//x = x_1x_2 \dots x_m$
2: Completar el último bloque con una secuencia de bits 0 si es necesario $//$*padding* del mensaje
3: Añadir un bloque extra que contenga la longitud del mensaje original módulo t $//$*length-block*
4: $h_0 = 0_1 0_2 \dots 0_k$ // Iniciación del resumen
5: **for** $i = 1$ **to** $m + 1$ **hacer**
6: $\quad h_i = f(h_0, x_i)$
7: **FinPara**
8: **Devolver** h_i

MDx

Una família de funciones resumen que fue ampliamente utilizada y basada en el esquema de Damgard-Merkle es la familia *Message Digest* (MD) cuyo primer algoritmo MD2 fue propuesto por Ronald Rivest en 1989. Esta primera versión prouesta para ordenadores de 8 bits se reformula con MD4 y posteriormente con MD5. Todas estas funciones producen resúmenes de 128 bits.

Pese a que todos los algoritmos de la familia se consideran débiles, pueden considerarse como versiones simplificadas de algoritmos actuales.

Algoritmo 2 Preproceso MD4/5 de un mensaje

Entrada: Mensaje x (binario) de longitud arbitraria n.
Salida: Mensaje x' (binario) de longitud múltiplo de 512.

1: Obtener x' extendiendo el mensaje (añadiendo un 1 seguido de suficientes 0s) para que su longitud sea congruente con 448 módulo 512 $//$*padding*
2: Añadir a x' la representación binaria del mensaje módulo 2^{64} (los 64 bits menos significativos) $//$*length-block*

El algoritmo MD4 modifica ligeramente el padding propuesto por Damgard-Merkle, sustituyendo el último bloque de 64 bits (length-block) por los bits menos significativos del mensaje original. Una vez preprocesado el mensaje (Algoritmo 2), este puede procesarse en bloques de 32 palabras de 16 bits, de acuerdo con el Algoritmo 3.

En esencia, el algoritmo considera un estado inicial formado por cuatro palabras de 16 bits (A, B, C y D) que modifica sucesivamente considerando cada uno de los bloques. El procesado de cada bloque supone la ejecución de un conjunto de *rounds*. Los rounds

MD4 se describen en los Algoritmos 4, 5 y 6.

Algoritmo 3 Resumen MD4 de un mensaje

Entrada: Mensaje x (binario) de longitud n múltiplo de 512.
Salida: $MD4(x)$ //Resumen de 128 bits

1: $A = 67452301_{hex}$
2: $B = EFCDAB89_{hex}$
3: $C = 98BADCFE_{hex}$
4: $D = 10325476_{hex}$
5: **Para toda** bloque de 512 bits **hacer**
6: Dividir el $Bloque_i$ en 16 palabras de 32 bits $Bloque_i = X[0], X[1], \ldots, X[15]$
7: $AA = A$
8: $BB = B$
9: $CC = C$
10: $DD = D$
11: **Round 1**
12: **Round 2**
13: **Round 3**
14: $A = A + AA$
15: $B = B + BB$
16: $C = C + CC$
17: $D = D + DD$
18: **FinPara**
19: **Devolver** $ABCD$

Muy pronto se desarrollan ataques a MD4 que no ejecutaban la primera o tercera vuelta. Actualmente se considera que MD4 está roto para cualquier uso.

MD5 preprocesa el mensaje del mismo modo que MD4. El algoritmo MD5 sigue básicamente el esquema MD4 ejecutando un cuarto round adicional y donde cada uno de los rounds ejecuta 20 operaciones en lugar de las 16 de MD4.

Pese a que MD5 no se considera seguro para uso criptográfico, MD5 sigue siendo fiable como medio para validar la integridad bien de ficheros propios cuyo contenido conocemos, bien de ficheros disponibles en la web.

Algoritmo 4 MD4: Round 1

1: $f(X,Y,Z) = (X\ AND\ Y)\ OR\ (\overline{X}\ AND\ Z)$ // Si X entonces Y, si no Z
2: $A = (A + f(B,C,D) + X[0]) <<< 3$
3: $D = (D + f(A,B,C) + X[1]) <<< 7$
4: $C = (C + f(D,A,B) + X[2]) <<< 11$
5: $B = (B + f(C,D,A) + X[3]) <<< 19$
6: $A = (A + f(B,C,D) + X[4]) <<< 3$
7: $D = (D + f(A,B,C) + X[5]) <<< 7$
8: $C = (C + f(D,A,B) + X[6]) <<< 11$
9: $B = (B + f(C,D,A) + X[7]) <<< 19$
10: $A = (A + f(B,C,D) + X[8]) <<< 3$
11: $D = (D + f(A,B,C) + X[9]) <<< 7$
12: $C = (C + f(D,A,B) + X[10]) <<< 11$
13: $B = (B + f(C,D,A) + X[11]) <<< 19$
14: $A = (A + f(B,C,D) + X[12]) <<< 3$
15: $D = (D + f(A,B,C) + X[13]) <<< 7$
16: $C = (C + f(D,A,B) + X[14]) <<< 11$
17: $B = (B + f(C,D,A) + X[15]) <<< 19$

Algoritmo 5 MD4: Round 2

1: $g(X,Y,Z) = (X\ AND\ Y)\ OR\ (X\ AND\ Z)\ OR\ (Y\ AND\ Z)$ // función mayoría
2: $A = (A + g(B,C,D) + X[0] + 5A827999) <<< 3$
3: $D = (D + g(A,B,C) + X[4] + 5A827999) <<< 7$
4: $C = (C + g(D,A,B) + X[8] + 5A827999) <<< 11$
5: $B = (B + g(C,D,A) + X[12] + 5A827999) <<< 19$
6: $A = (A + g(B,C,D) + X[1] + 5A827999) <<< 3$
7: $D = (D + g(A,B,C) + X[5] + 5A827999) <<< 7$
8: $C = (C + g(D,A,B) + X[9] + 5A827999) <<< 11$
9: $B = (B + g(C,D,A) + X[13] + 5A827999) <<< 19$
10: $A = (A + g(B,C,D) + X[2] + 5A827999) <<< 3$
11: $D = (D + g(A,B,C) + X[6] + 5A827999) <<< 7$
12: $C = (C + g(D,A,B) + X[10] + 5A827999) <<< 11$
13: $B = (B + g(C,D,A) + X[14] + 5A827999) <<< 19$
14: $A = (A + g(B,C,D) + X[3] + 5A827999) <<< 3$
15: $D = (D + g(A,B,C) + X[7] + 5A827999) <<< 7$
16: $C = (C + g(D,A,B) + X[11] + 5A827999) <<< 11$
17: $B = (B + g(C,D,A) + X[15] + 5A827999) <<< 19$

Algoritmo 6 MD4: Round 3

1: $h(X,Y,Z) = (X \oplus Y \oplus Z)$
2: $A = (A + h(B,C,D) + X[0] + 6ED9EBA1) <<< 3$
3: $D = (D + h(A,B,C) + X[8] + 6ED9EBA1) <<< 9$
4: $C = (C + h(D,A,B) + X[4] + 6ED9EBA1) <<< 11$
5: $B = (B + h(C,D,A) + X[12] + 6ED9EBA1) <<< 15$
6: $A = (A + h(B,C,D) + X[2] + 6ED9EBA1) <<< 3$
7: $D = (D + h(A,B,C) + X[10] + 6ED9EBA1) <<< 9$
8: $C = (C + h(D,A,B) + X[6] + 6ED9EBA1) <<< 11$
9: $B = (B + h(C,D,A) + X[14] + 6ED9EBA1) <<< 15$
10: $A = (A + h(B,C,D) + X[1] + 6ED9EBA1) <<< 3$
11: $D = (D + h(A,B,C) + X[9] + 6ED9EBA1) <<< 9$
12: $C = (C + h(D,A,B) + X[5] + 6ED9EBA1) <<< 11$
13: $B = (B + h(C,D,A) + X[13] + 6ED9EBA1) <<< 15$
14: $A = (A + h(B,C,D) + X[3] + 6ED9EBA1) <<< 3$
15: $D = (D + h(A,B,C) + X[11] + 6ED9EBA1) <<< 9$
16: $C = (C + h(D,A,B) + X[7] + 6ED9EBA1) <<< 11$
17: $B = (B + h(C,D,A) + X[15] + 6ED9EBA1) <<< 15$