MD5

En criptografía, MD5 (abreviatura de *Message-Digest Algorithm 5*, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado.

Historia

MD5 es uno de los algoritmos de reducción criptográficos diseñados por el profesor Ronald Rivest del MIT (*Massachusetts Institute of Technology*, Instituto Tecnológico de Massachusetts). Fue desarrollado en 1991 como reemplazo del algoritmo MD4 después de que Hans Dobbertin descubriese su debilidad.
A pesar de su amplia difusión actual, la sucesión de problemas de seguridad detectados desde que, en 1996, Hans Dobbertin anunciase una colisión de *hash* plantea una serie de dudas acerca de su uso futuro.
Codificación [editar]
La codificación del MD5 de 128 bits es representada típicamente como un número de 32 dígitos hexadecimal. El siguiente código de 28 bytes ASCII será tratado con MD5 y veremos su correspondiente *hash* de salida:

- MD5("Esto sí es una prueba de MD5") = e99008846853ff3b725c27315e469fbc
- Un simple cambio en el mensaje nos da un cambio total en la codificación *hash*, en este caso cambiamos dos letras, el «sí» por un «no».
- MD5("Esto no es una prueba de MD5") = dd21d99a468f3bb52a136ef5beef5034
- Otro ejemplo sería la codificación de un campo vacío:
- MD5("") = d41d8cd98f00b204e9800998ecf8427e

Algoritmo

Terminologías y notaciones
En este documento "palabra" es una entidad de 32 bits y *byte* es una entidad de 8 bits. Una secuencia de bytes puede ser interpretada de manera natural como una secuencia de bits, donde cada grupo consecutivo de ocho bits se interpreta como un byte con el bit más significativo al principio. Similarmente, una secuencia de bytes puede ser interpretada como una secuencia de 32 bits (palabra), donde cada grupo consecutivo de cuatro bytes se interpreta como una palabra en la que el byte menos significativo está al principio.
El símbolo "+" significa suma de palabras.

X <<< s se interpreta por un desplazamiento a la izquierda 's' posiciones
not(x) se entiende como el complemento de x

Descripción del algoritmo md5

Empezamos suponiendo que tenemos un mensaje de 'b' bits de entrada, y que nos gustaría encontrar su resumen. Aquí 'b' es un valor arbitrario entero no negativo, pero puede ser cero, no tiene por qué ser múltiplo de ocho, y puede ser muy largo. Imaginemos los bits del mensaje escritos así:

m0 m1 ... m{b-1}

Los siguientes cinco pasos son efectuados para calcular el resumen del mensaje.

Paso 1. Añadiendo bits

El mensaje será extendido hasta que su longitud en bits sea congruente con 448, módulo 512. Esto es, si se le resta 448 a la longitud del mensaje tras este paso, se obtiene un múltiplo de 512. Esta extensión se realiza siempre, incluso si la longitud del mensaje es ya congruente con 448, módulo 512.
La extensión se realiza como sigue: un sólo bit "1" se añade al mensaje, y después bits "0" se añaden hasta que la longitud en bits del mensaje extendido se haga congruente con 448, módulo 512. En todos los mensajes se añade al menos un bit y como máximo 512.

Paso 2. Longitud del mensaje

Un entero de 64 bits que represente la longitud 'b' del mensaje (longitud antes de añadir los bits) se concatena al resultado del paso anterior. En el supuesto no deseado de que 'b' sea mayor que 2^64, entonces sólo los 64 bits de menor peso de 'b' se usarán.
En este punto el mensaje resultante (después de rellenar con los bits y con 'b') se tiene una longitud que es un múltiplo exacto de 512 bits. A su vez, la longitud del mensaje es múltiplo de 16 palabras (32 bits por palabra). Con M[0 ... N-1] denotaremos las palabras del mensaje resultante, donde N es múltiplo de 16.

Paso 3. Inicializar el búfer MD

Un búfer de cuatro palabras (A, B, C, D) se usa para calcular el resumen del mensaje. Aquí cada una de las letras A, B, C, D representa un registro de 32 bits. Estos registros se inicializan con los siguientes valores hexadecimales, los bits de menor peso primero:

palabra A: 01 23 45 67
palabra B: 89 ab cd ef
palabra C: fe dc ba 98
palabra D: 76 54 32 10

Paso 4. Procesado del mensaje en bloques de 16 palabras

Primero definimos cuatro funciones auxiliares que toman como entrada tres palabras de 32 bits y su salida es una palabra de 32 bits.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Los operadores $\oplus, \wedge, \vee, \neg$son las funciones XOR, AND, OR y NOT respectivamente.
En cada posición de cada bit F actúa como un condicional: si X, entonces Y sino Z. La función F podría haber sido definida usando + en lugar de v ya que XY y not(x) Z nunca tendrán unos ('1') en la misma posición de bit. Es interesante resaltar que si los bits de X, Y y Z son independientes y no sesgados, cada uno de los bits de F(X,Y,Z) será independiente y no sesgado.

Las funciones G, H e I son similares a la función F, ya que actúan "bit a bit en paralelo" para producir sus salidas de los bits de X, Y y Z, en la medida que si cada bit correspondiente de X, Y y Z son independientes y no sesgados, entonces cada bit de

G(X,Y,Z), H(X,Y,Z) e I(X,Y,Z) serán independientes y no sesgados. Nótese que la función H es la comparación bit a bit "xor" o función "paridad" de sus entradas.
Este paso usa una tabla de 64 elementos T[1 ... 64] construida con la función Seno. Denotaremos por T[i] el elemento i-ésimo de esta tabla, que será igual a la parte entera del valor absoluto del seno de 'i' 4294967296 veces, donde 'i' está en radianes.

Código del MD5:

```
 /* Procesar cada bloque de 16 palabras. */
para i = 0 hasta N/16-1 hacer

   /* Copiar el bloque 'i' en X. */
   para j = 0 hasta 15 hacer
     hacer X[j] de M[i*16+j].
   fin para /* del bucle 'j' */

 /* Guardar A como AA, B como BB, C como CC, y D como DD. */
 AA = A
 BB = B
 CC = C
 DD = D

 /* Ronda 1. */
 /* [abcd k s i] denotarán la operación
      a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  0  7  1] [DABC  1 12  2] [CDAB  2 17  3] [BCDA  3 22  4]
 [ABCD  4  7  5] [DABC  5 12  6] [CDAB  6 17  7] [BCDA  7 22  8]
 [ABCD  8  7  9] [DABC  9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
 [ABCD 12  7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

 /* Ronda 2. */
 /* [abcd k s i] denotarán la operación
     a = b + ((a + G(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  1  5 17] [DABC  6  9 18] [CDAB 11 14 19] [BCDA  0 20 20]
 [ABCD  5  5 21] [DABC 10  9 22] [CDAB 15 14 23] [BCDA  4 20 24]
 [ABCD  9  5 25] [DABC 14  9 26] [CDAB  3 14 27] [BCDA  8 20 28]
 [ABCD 13  5 29] [DABC  2  9 30] [CDAB  7 14 31] [BCDA 12 20 32]

 /* Ronda 3. */
 /* [abcd k s t] denotarán la operación
     a = b + ((a + H(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
 [ABCD  5  4 33] [DABC  8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
 [ABCD  1  4 37] [DABC  4 11 38] [CDAB  7 16 39] [BCDA 10 23 40]
 [ABCD 13  4 41] [DABC  0 11 42] [CDAB  3 16 43] [BCDA  6 23 44]
 [ABCD  9  4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA  2 23 48]

 /* Ronda 4. */
 /* [abcd k s t] denotarán la operación
     a = b + ((a + I(b, c, d) + X[k] + T[i]) <<< s). */
 /* Hacer las siguientes 16 operaciones. */
```

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

/* Ahora realizar las siguientes sumas. (Este es el incremento de cada
   uno de los cuatro registros por el valor que tenían antes de que
   este bloque fuera inicializado.) */

   A = A + AA
   B = B + BB
   C = C + CC
   D = D + DD

fin para /* del bucle en 'i' */

Paso 5. Salida

El resumen del mensaje es la salida producida por A, B, C y D. Esto es, se comienza el byte de menor peso de A y se acaba con el byte de mayor peso de D.

Seguridad

A pesar de haber sido considerado criptográficamente seguro en un principio, ciertas investigaciones han revelado vulnerabilidades que hacen cuestionable el uso futuro del MD5. En agosto de 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu anunciaron el descubrimiento de colisiones de *hash* para MD5. Su ataque se consumó en una hora de cálculo con un clúster IBM P690.

Aunque dicho ataque era analítico, el tamaño del *hash* (128 bits) es lo suficientemente pequeño como para que resulte vulnerable frente a ataques de «fuerza bruta» tipo «cumpleaños». El proyecto de computación distribuida *MD5CRK* arrancó en marzo de 2004 con el propósito de demostrar que MD5 es inseguro frente a uno de tales ataques, aunque acabó poco después del aviso de la publicación de la vulnerabilidad del equipo de Wang.

Debido al descubrimiento de métodos sencillos para generar colisiones de *hash*, muchos investigadores recomiendan su sustitución por algoritmos alternativos tales como SHA-1 o RIPEMD-160.

Aplicaciones

Los resúmenes MD5 se utilizan extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de Internet no se ha alterado. Comparando una suma MD5 publicada con la suma de comprobación del archivo descargado, un usuario puede tener la confianza suficiente de que el archivo es igual que el publicado por los desarrolladores. Esto protege al usuario contra los 'Caballos de Troya' o 'Troyanos' y virus que algún otro usuario malicioso pudiera incluir en el software. La comprobación de un archivo descargado contra su suma MD5 no detecta solamente los archivos alterados de una manera maliciosa, también reconoce una descarga corrupta o incompleta.

Para comprobar la integridad de un archivo descargado de Internet se puede utilizar una herramienta MD5 para comparar la suma MD5 de dicho archivo con un archivo MD5SUM con el resumen MD5 del primer archivo. En los sistemas UNIX, el comando de *md5sum* es un ejemplo de tal herramienta. Además, también está implementado en el lenguaje de *scripting* PHP como MD5("") entre otros.

En sistemas UNIX y GNU/Linux se utiliza el algoritmo MD5 para calcular el *hash* de las claves de los usuarios. En el disco se guarda el resultado del MD5 de la clave que se introduce al dar de alta un usuario, y cuando éste quiere entrar en el sistema se compara el *hash* MD5 de la clave introducida con el *hash* que hay guardado en el disco duro. Si coinciden, es la misma clave y el usuario será autenticado.
El MD5 también se puede usar para comprobar que los correos electrónicos no han sido alterados usando claves públicas y privadas.
Véase también [editar]
MD2
MD4
SHA
Tiger
RIPEMD-160
WHIRLPOOL
CRC
Hash

MD5 EN PHP

Hoy en día la mayoría de las páginas web utilizan bases de datos para poder desarrollar portales dinámicos y así hacerlos más atractivos a la vez que útiles. Pero esta información que se guarda en la base de datos tiene que tener algún tipo de protección. Es por ello que algunos campos se guardan encriptados en la base de datos, principalmente cuando una página requiere el nombre de usuario y contraseña, esta última se encripta y se guarda en la Base de datos.
En PHP se utiliza la función MD5 (Message Digest 5), que es una función hash irreversible (de un sólo sentido) , es decir, encripta el password tecleado por el usuario y es imposible que partiendo desde la cadena encriptada se vuelva a la contraseña origen. Por esto mismo no hay problema de que alguien pueda acceder al campo encriptado de la base de datos.
Como en la base de datos se guarda la contraseña encriptada, cuando un usuario quiere acceder, habrá que realizar una comparación entre el password que introduce encriptado en MD5, y lo que tenemos en la base de datos, (que es la contraseña encriptada en MD5), si coincide se le permite el acceso, si no, se rechaza.
MD5 se utiliza también para que cuando un usuario olvida su password, si quiere recuperar la contraseña se le pide que introduzca por ejemplo el correo, y se le envía un mail con una URL tal que si entra en ella genere una nueva contraseña que se le indica al usuario y se reescribe en md5 en la base de datos (borrando la anterior contraseña).
Hay que tener en cuenta que esto no es 100% seguro, puesto que la contraseña se encripta en el servidor, entonces al enviar la contraseña desde el cliente al servidor podría ser interceptada.
Para hacernos una idea, el algoritmo MD5 convierte el mensaje en un bloque múltiplo de 512 bits, (si hace falta añadirá bits por el final). Luego coge el primer bloque de 512 bits del mensaje y realiza diversas operaciones lógicas con los 128 bits de cuatro vectores iniciales ABCD de 32 bits cada uno. (Dichos vectores tendrán el valor inicial que nosotros queramos).
Como resultado obtiene una salida de 128 bits que se convierte en el nuevo conjunto de los 4 vectores ABCD. Se repite el algoritmo hasta procesar el último bloque del mensaje. Al terminar, el algoritmo devuelve los últimos 128 bits de estas operaciones.
La definición de la función md5 en PHP es:
string md5 (string cad).

MD5 EN JAVA

En muchas ocasiones nos encontramos con la necesidad de encriptar una cadena o verificar la integridad de un archivo transmitido desde un punto a otro.
Una de las alternativas es utilizar el algoritmo MD5.
Para utilizarlo en JAVA tenemos varias maneras, una de ellas la presento a continuacion:

```
/* Cifrar una cadena */
public String getMD5(String passwd){
byte[] textBytes = passwd.getBytes();
MessageDigest md = null;
try {
md = MessageDigest.getInstance("MD5");
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();
}
md.update(textBytes);
byte[] codigo = md.digest();
md5 = this.convertToHex(md5sum);
}
/* Convierta a Hexa */
private String convertToHex(byte[] data) {
StringBuffer buf = new StringBuffer();
for (int i = 0; i &lt; data.length; i++) {
int halfbyte = (data[i] &gt;&gt;&gt; 4) &amp; 0×0F;
int two_halfs = 0;
do {
if ((0 &lt;= halfbyte) &amp;&amp; (halfbyte &lt;= 9))
buf.append((char) ('0' + halfbyte));
else
buf.append((char) ('a' + (halfbyte - 10)));
halfbyte = data[i] &amp; 0×0F;
} while(two_halfs++ &lt; 1);
}
return buf.toString();
/* Calcular el MD5 de un archivo*/
}
public String getMD5(File archivo) {
byte[] textBytes = new byte[10000];
MessageDigest md = null;
int read = 0;
String md5 = null;
try {
InputStream is = new FileInputStream(archivo);
md = MessageDigest.getInstance("MD5");
while ((read = is.read(textBytes)) &gt; 0) {
md.update(textBytes, 0, read);
}
is.close();
byte[] md5sum = md.digest();
md5 = this.convertToHex(md5sum);
} catch (FileNotFoundException e1) {
```

```
e1.printStackTrace();
} catch (NoSuchAlgorithmException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
return md5;
}
```