inTechractive.com

Funciones Hash:

Aplicaciones en los Sistemas Computacionales

Elaborado por: Gimer A. Cervera Evia, Ph.D.

Funciones Hash: Aplicaciones en los Sistemas Computacionales.

¿Qué es una función hash? ¿Cómo funciona y cuáles son sus aplicaciones en los sistemas computacionales?

¿Qué es una función hash?

Una función Hash (H) o función de un solo sentido (i.e., one-way function) es un algoritmo matemático aplicado a un bloque de datos para convertirlo en una nueva cadena de caracteres de longitud fija. Su implementación en el área de seguridad computacional es bastante común. Recientemente, esta tecnología es también aplicada para garantizar la integridad de transacciones financieras. Estas funciones son utilizadas en la tecnología detrás de las cadenas de bloques en el área financiera (i.e., Blockchains), la comunicación P2P (i.e., peer-two-peer), la descarga de archivos a través de servidores Torrent, firmas digitales, autenticación de usuarios, etc.

Aspectos Técnicos

Los algoritmos usados en las funciones Hash están fuera del alcance de este artículo. Sin embargo, vamos a comentar los requisitos y propiedades que deben de cumplir. Para que una función hash (H) sea válida debe de cumplir con los siguientes requisitos [1]:

- La función H puede ser aplicada a un mensaje ***m*** de cualquier tamaño, ver figura 1.
- H(m) es conocido como el valor hash de m.
- La función H es siempre conocida.
- H(m) siempre produce una salida con la misma longitud, para cualquier valor de m.
- H(m) debe ser relativamente fácil de calcular, para cualquier valor de m.
- Si m cambia en al menos un bit, H(m) genera un resultado diferente de igual longitud, ver figura 1.

Para que una función hash funcione correctamente, esta debe de cumplir con tres importantes propiedades [1]:

1. No es posible conocer el mensaje original a partir de la cadena resultante. Es decir, si se conoce H(m), no debe ser posible obtener el mensaje original, es decir, m. (i.e., one-way property). Por lo tanto, si se tiene que H(m) produce la cadena resultante y. No es posible inferir el valor original m a partir de y.
2. Dado que se conoce una cadena de caracteres obtenida de calcular H(m). Es computacionalmente imposible encontrar otro mensaje n tal que al aplicar la misma

función hash se obtenga el mismo resultado. Es decir, si n ≠ m entonces H(n) ≠ H(m). (i.e., weak collision resistance).

3. Es computacionalmente imposible encontrar un par de valores diferentes n y m, tal que H(n) = H(m), (i.e., strong collision resistance).

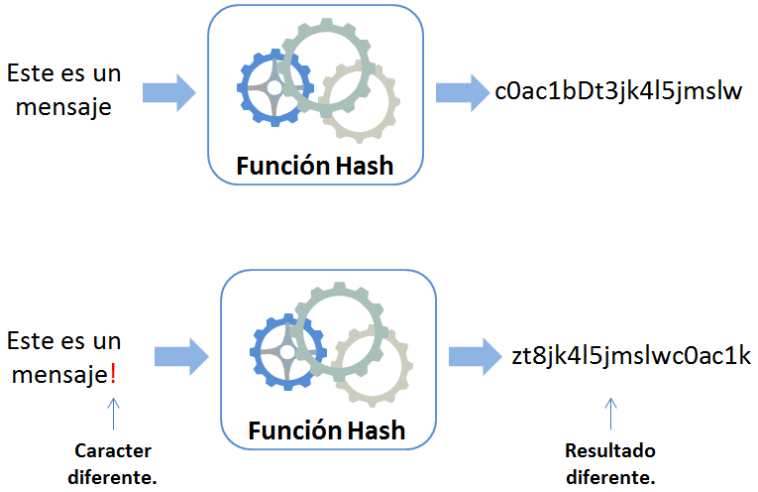

Figura 1. Resultado de aplicar una función Hash a un mensaje.

Este mecanismo funciona de la siguiente manera: vamos a suponer que se tiene un mensaje m y la función hash H. El mensaje original se convierte a su equivalente en bits para obtener una cadena a la cual se le aplica la función hash, es decir, H(m).

m = "Este es un mensaje" → 10101010110001101010

Supongamos que m se convierte a su equivalente en bits: 10101010110001101010, entonces:

H(m) = H(10101010110001101010)

Esta función va a tener como resultado otra cadena de caracteres que pueden ser de tipo alfanumérico, es decir, una combinación de caracteres y dígitos, por ejemplo:

H(10101010110001101010) → f1b9qk&cdoWpla83kd

Esta técnica garantiza que si la cadena original se altera en al menos un bit, el resultado es una cadena significativamente distinta. Es decir, si se tiene un mensaje

m' = 101010101100011010**1** **¡Nuevo bit!**

H(m') = 3oXtla8r22df1b9qklc → **¡Resultado diferente!**

H(m) ≠ H(m')

f1b9qk&cdoWpla83kd ≠ 3oXtla8r22df1b9qklc

A continuación vamos a describir su aplicación en la tecnología que está detrás de los programas de descarga de archivos Torrent y el esquema de autenticación de usuarios de Leslie Lamport.

Aplicaciones de las funciones hash - Torrents

La tecnología Torrent es un esquema colaborativo para compartir información. Generalmente archivos de gran tamaño. Este mecanismo permite que los usuarios compartan y descarguen diferentes partes de un mismo conjunto de datos de distintos sitios para no depender de un solo servidor. Todos los usuarios del servicio se convierten en servidores durante o al completar la descarga de un archivo.

Esta técnica funciona de la siguiente manera. El servidor Torrent divide el archivo en *n-bloques* (i.e., b_1, b_2, …, b_n) que se distribuyen en *k-servidores*. El servidor Torrent calcula el valor hash de cada una de esos bloques, es decir, $H(b_i) = H_i$, veamos el ejemplo de la figura 2.

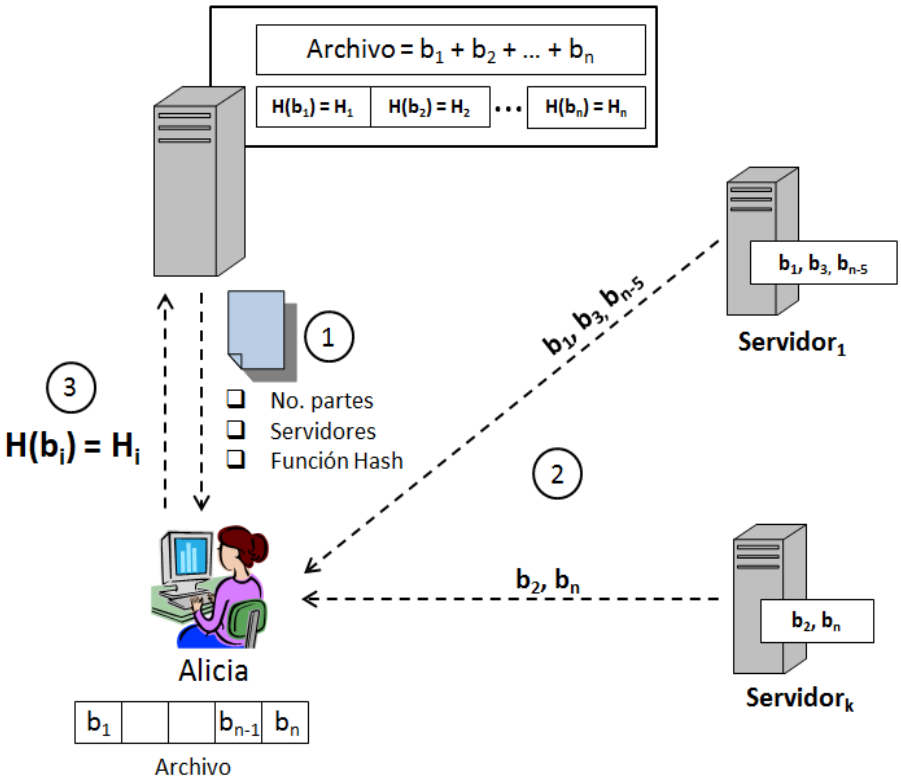

Figura 2. Proceso de descarga en un servidor Torrent.

1. Alicia descarga del servidor principal el número de bloques que debe de recolectar, la dirección de los servidores que tienen los bloques del archivo y la función hash que se debe de utilizar (Paso 1). El valor hash de cada bloque puede estar en el servidor o localmente en el equipo de Alicia.

2. Alicia solicita los bloques a los servidores involucrados y dependiendo de su disponibilidad estos le envían a Alicia los bloques que tienen. El orden de recepción de los bloques no es importante (Paso 2).

3. Cada vez que Alice termina de recibir un bloque, se le aplica la función hash y se verifica que el resultado concuerde con el valor hash previamente calculado por el servidor Torrent (Paso 3). La única forma de que Alicia obtenga el valor correcto H_i es recibiendo el bloque b_i de forma íntegra. Si el bloque está corrupto, es decir $\mathbf{b'_i}$, se obtiene otro valor al aplicar la función hash, es decir: $\mathbf{H(b'_i) \neq H(b'_i)}$.

   En este caso, Alicia rechaza ese bloque e intenta descargarlo de nuevo. Las propiedades de las funciones hash nos permiten garantizar la integridad de la información.

4. El proceso concluye cuando Alice ha recibido y validado los *n* bloques que conforman el archivo.

Autenticación de Lamport

El esquema de autenticación diseñado por Leslie Lamport (1981)[2] es otro ejemplo de la aplicación de las cadenas de Hash. Cualquier sistema computacional debe de almacenar en su base de datos las contraseñas cifradas. Esto es, el servidor no conoce las contraseñas en texto plano para ofrecer mayor seguridad a los usuarios. Si algún atacante tiene acceso a la base de datos, este no podrá conocer la contraseña original. Por otra parte, cada vez que un usuario se autentica, la información almacenada en la base de datos debe de actualizarse. Esto con la finalidad de prevenir que no se pueda reutilizar la información empleada en un proceso de autenticación previo. Tenemos que tomar en cuenta que es posible perpetrar ataques aun sin conocer el texto original (e.g., replay attack).

En la figura x, se presenta un ejemplo simple de su uso e implementación. Supongamos que Alicia se desea autenticar en un servidor. Para ello va a necesitar su cuenta de usuario y contraseña. Al momento de registrarse el servidor almacena en la base de datos la contraseña cifrada utilizando una función hash (H). Esta función se aplica n número de veces, es decir, se aplica un hash sobre un valor hash múltiples veces (ver etapa 1).

Para autentificarse, Alicia solicita acceso al servidor, este le responde con un reto, es decir, le envía un valor n. Alicia le responde al servidor su contraseña aplicando la función hash n-1 veces. El servidor recibe el valor enviado por Alicia y le aplica la función hash una vez más, es decir:

$$\mathbf{H(H^{n-1}(password)) = H^{n}(password)}$$

Si el resultado coincide con el valor almacenado en la base de datos, entonces Alicia recibe una llave de sesión (i.e., $Session_k$) por parte del servidor para continuar con las transacciones. Al mismo tiempo, el servidor almacena en su base de datos el valor:

$$\mathbf{H^{n-1}(password)}$$

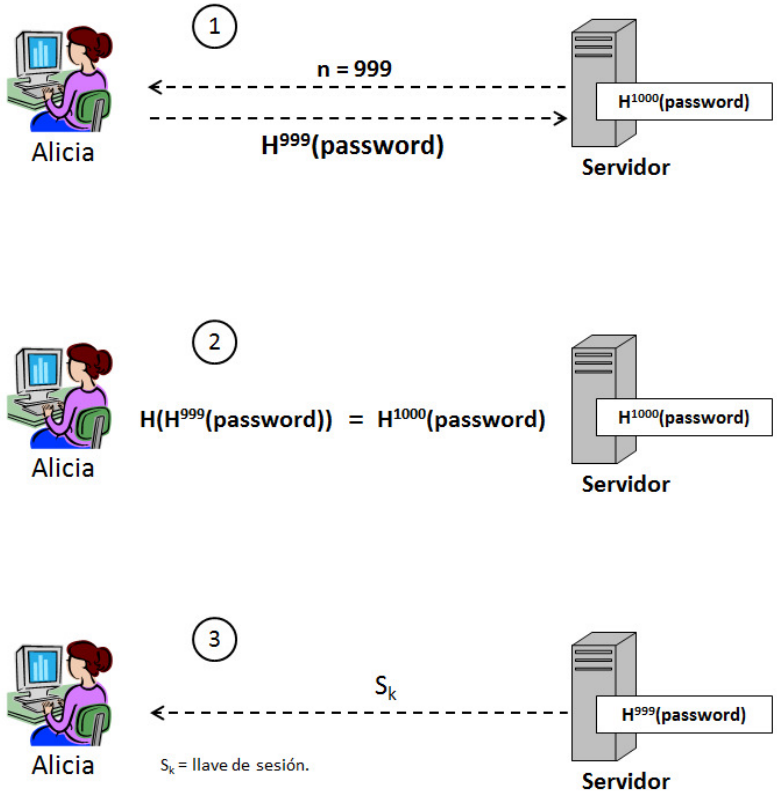

Figura 3. Esquema de autenticación de Lamport usando funciones Hash.

Finalmente, la información almacenada en la base de datos es actualizada. Para autenticarse de nuevo, Alicia ahora deberá generar el valor: $\mathbf{H^{n-2}(password)}$

Gracias a las propiedades de las funciones Hash, es imposible que el atacante pueda calcular el valor de $\mathbf{H^{n-2}(password)}$ sin conocer el password original. Incluso conociendo la función hash utilizada, el valor de n y valores hash previos como:

$$\mathbf{H^{n}(password)} \text{ y } \mathbf{H^{n-1}(password).}$$

El uso de funciones hash como mecanismo de autenticación es combinada con otras técnicas criptográficas para evitar otro tipo de ataques, por ejemplo, el uso de marcas de tiempo (i.e., time-stamps) o criptografía con llaves públicas y privadas.

Conclusión

Las funciones Hash son un mecanismo de cifrado utilizado en muchas áreas en ciencias computacionales, principalmente en la seguridad informática para garantizar la integridad de la información o la autenticación de usuarios.

Referencias:

[1] William Stallings. 2006. *Cryptography and Network Security: Principles and Practice* (4th ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.

[2] Leslie Lamport. 1981. Password authentication with insecure communication. Commun. ACM 24, 11 (November 1981), 770-772. DOI=http://dx.doi.org/10.1145/358790.358797