SHA-256

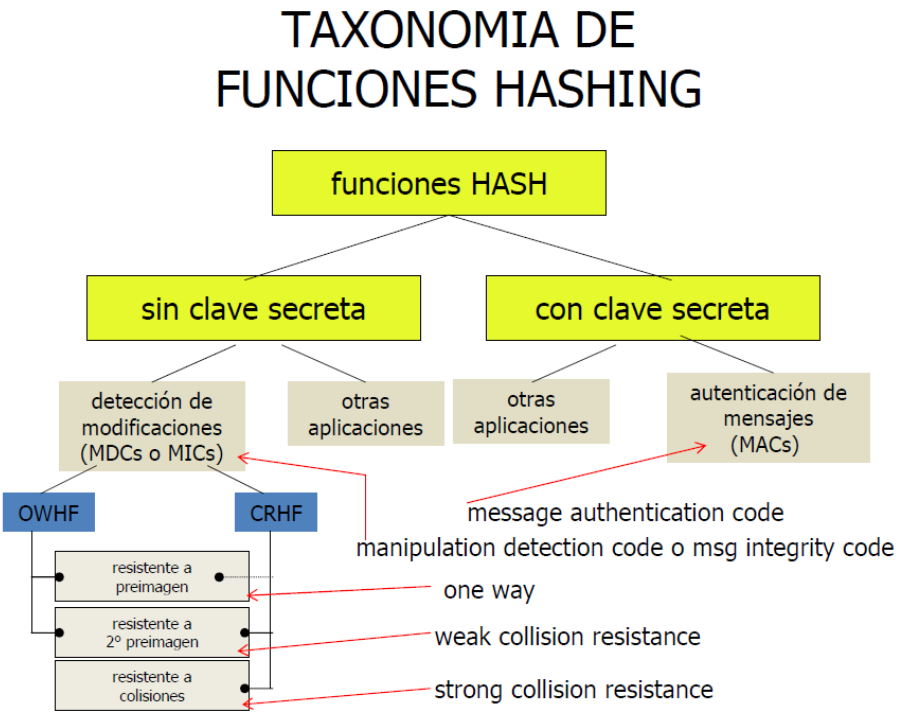
Autores:
Juan Mas Aguilar
Jose Javier Cortés Tejada
Ignacio Sande Soltero

Introducción

Las funciones hash criptográficas son equivalentes a sus homónimas en estructuras de datos con la salvedad de que las primeras están especialmente preparadas para su uso en la criptografía. Esto las hace especialmente resistentes a los ataques y son idóneas para la seguridad de los sistemas informáticos. Cuando usamos o desarrollamos una función hash criptográfica, buscamos que cumplan con una serie de propiedades:

- Compresión: h mapea una entrada x de tamaño arbitrario en una salida $h(x)$ de logitud fija de n-bits.
- Facibilidad de computo: dado h y una entrada x, $h(x)$ es fácilmente computable.
- Resistencia a la preimagen (**ONE WAY HASH FUNCION**): para prácticamente cualquier entrada es computacionalmente irrealizable encontrar un x (desconocido) tal que $h(x) = y$ (conocido).
- Resistencia a la segunda preimagen (**OWHF: WEAK ONE WAY HASH FUNCTION**): dados x y su correspondiente $y = h(x)$, es computacionalmente irrealizable encontrar otro $x^{'}$ distinto a x tal que $y' = h(x^{'}) = y$.
- Resistencia a las colisiones (**CRHF: STRONG ONE WAY HASH FUNCTION**): es computacionalmente irrealizable hallar dos entradas (x , $x^{'}$) distintas que posean la misma salida $h(x)$. (Esto difiere del anterior en que hay libre elección de los dos valores x y por eso es más fuerte).
- No correlación: los bits de entrada y los bits de salida no deben estar correlacionados. Vinculado con esto, es deseable un efecto de avalancha en la cual cada bit de entrada afecta a cada bit de salida.
- Resistencia a la casi colisión: debe ser difícil obtener dos entradas (x, $x^{'}$) para las cuales sus imágenes (y, $y^{'}$) difieran solamente en unos pocos bits.
- Resistencia a las preimagenes parciales: debe ser tan difícil recuperar una parte de la preimagen de un hash como recuperar su totalidad.

Las funciones hash criptográficas tienen multitud de usos en la actualidad, pero, es aceptada la división en dos tipos: MAC y MDC. En el primero el uso de la función hash consiste en el calculo de un código de autentificación de mensaje. En el segundo tipo, el uso de la función consiste en comprobar la integridad de la clave (por ejemplo, integridad de pruebas informáticas judiciales).



Otra característica de las funciones hash es la longitud del mensaje inicial, que debe tener una longitud mínima para garantizar su seguridad (160 bits). Esto se garantiza con la llamada paradoja del cumpleaños que determina la probabilidad de colisión en un mensaje, en base a la siguiente fórmula:

$$(1-\frac{1}{n})(1-\frac{2}{n})...(1-\frac{k-1}{n}) = \prod_{i=1}^{k-1}(1-\frac{i}{n})$$

Familia SHA

La familia SHA (Secure Hash Algorithm) es un conjunto de funciones relacionadas publicadas por el Instituto Nacional de Estandares y Tecnología (NIST). Esta familia de funciones son de tipo MDC, es decir, están pensadas especialmente para el guardado y autenticación de secretos.

Incialmente, se publicó el algoritmo SHA-0 pero se descubrieron vulnerabilidades a dicho algoritmo y se desarrolló una nueva versión llamada SHA-1 (ambos producen una salida resumen de 160 bits (20 bytes) de un mensaje que puede tener un tamaño máximo de 264 bits). Dicho algoritmo aun está en uso pero se descubrió una vulnerabilidad que disminuía la dificultad de encontrar una colisión debido a la divulgación de varias funciones criptográficas. No obstante, dicha dificultad aun es de 2^{69}, hoy por hoy considerada una cantidad de computo imposible.

Aun así, a pesar de que es computacionalmente imposible calcular una colisión usando SHA-1, la NSA y el NIST consideraron necesario el desarrollo de una nueva versión del algoritmo, SHA-2. Esta versión es la considerada actualmente como el estándar. En especial, el algoritmo SHA-256 es uno de los más usados a nivel mundial.

SHA-256

SHA-256 (Secure Hash Algorithm) es el algoritmo de hash principal usado en el protocol Bitcoin. Pertenece a la família de funciones de hash SHA-2 diseñadas por la NSA y aceptadas por el NIST como el estándar a usar en algoritmos de hash por la administración norteamericana, para sustituir a su predecesor más inmediato, SHA-1, del cual se conocen varias vulnerabilidades de seguridad.

Funcionamiento del algoritmo

SHA-256 opera de manera similar a MD4, MD5 y SHA-1, de tal manera que el mensaje que va a ser *hasheado*

1. es rellenado de tal manera que su longitud final resulte ser múltiplo de 512 y entonces
2. es parseado en bloques $M^{(1)}, M^{(2)}, ..., M^{(n)}$ de 512 bits.

A continuación los bloques del mensaje son procesador uno a uno, empezando con un valor inicial fijo para el hash $H^{(0)}$, el cual es computado secuencialmente $H^{(i)} = H^{(i-1)} + C_{M^{(i)}}(H^{(i-1)})$, donde C es la función de compresión de SHA-256 y + implica suma $mod\ 2^{32}$.

La función de compresión SHA-256 opera en un bloque de 512 bits y en un valor intermedio del hash de 256 bits. Es esencialmente un algoritmo cifrado de bloques de 512 bits que cifra el valor de hash intermedio utilizando el bloque de mensajes como clave. Por lo tanto allí tenemos dos componentes principales:

1. la función de compresión SHA-256.
2. la programación de mensajes SHA-256.

El valor inicial del hash $H^{(0)}$ es la siguiente secuencia de palabras de 32 bits (obtenidos mediante raices cuadradas de los primeros 8 números primos):

$$\begin{aligned} H_1^{(0)} &= 6a09e667 \\ H_2^{(0)} &= bb67ae85 \\ H_3^{(0)} &= 3c6ef372 \\ H_4^{(0)} &= a54ff53a \\ H_5^{(0)} &= 510e527f \\ H_6^{(0)} &= 9b05688c \\ H_7^{(0)} &= 1f83d9ab \\ H_8^{(0)} &= 5be0cd19 \end{aligned}$$

La computación del hash del mensaje se divide en dos partes:

1. Rellenamos el mensaje: suponiendo que la longitud del mensaje es M es l bits. Añadimos un bit 1 al final del mensaje y despues añadimos k bits, donde k es el menor solución no negativa a la ecuación $l + 1 + k$ es equivalente a 448 mod 512. A esto concatenamos el bloque 64 bits que codifica el número l escrito en binario.
2. Parseamos el mensaje en N bloques de 512 bits en $M^{(1)}, M^{(2)}, ..., M^{(n)}$ bloques. Los primeros 32 bits del bloque de mensaje i son denotados $M_0^{(i)}$, mientras que los 32 bits siguientes son denotados $M_1^{(i)}$.

   Tras estas operaciones se entra al bucle principal del algoritmo, donde para cada uno de los N bloques:

   a) Se copia el valor del hash.

   b) Aplicamos la función de compresión.

   c) Calculamos el nuevo valor del hash.

   d) $H^{(N)} = (H_1^{(N)}, H_2^{(N)}, ..., H_8^{(N)})$.

   e) Por último procesamos los bloques del mensaje en función del hash y obtenemos el resultado final.

Uso de SHA-256 en minería de bitcoin

El algoritmo de cifrado hash SHA-256 es muy usado en el sistema bitcoin. En especial, se usa para la creación de direcciones bitcoin (investigar) y la verificación de transacciones a través de pruebas de trabajo (proof of work) y árboles Merkel (Merkel Trees) usados para la protección del llamado block-chain (registro general de todas las transacciones de bitcoins).

Arboles de Merkle

Un árbol de Merkle (Merkle Tree) o árbol de hash, es una construcción en forma de árbol de valores en donde cada nodo interior (nodos que no son nodos finales) es el resultado de aplicar una función de hash (sea SHA-1, SHA-2, etc.) sobre el valor de los nodos hijo hasta llegar a un nodo raíz llamado Merkle root del árbol. Pueden ser binarios (cada nodo interior tiene dos descendientes) o no (cada nodo interior tiene un número determinado de descendientes).

La funcionalidad de estos algoritmos es la de permitir verificar de forma eficiente y segura la integridad e inclusión de todos los datos de estructuras de datos grandes. Además, se pueden considerar como una generalización de las cadenas de hash (hash chains), en donde la verificación de un elemento es proporcional al logaritmo del número de nodos del árbol, a diferencia de las cadenas de hash, en donde el coste es lineal con el número de hashes contenidos en la cadena.

Este tipo de árbol es usado en la mínería de bitcoins para proteger el conocido como block-chain. Dicha estructura es el registro de todas y cada una de las transacciones de bitcoins. Los bloques de este block-chain son generados por los *bitcoin miners*.

Algoritmo Hashcash

Bitcoin usa el algoritmo HashCash Proof-of-work como el nucleo de la minería. Todos los *bitcoin miners* ya sean referentes a CPU, GPU, FPGA o ASICs están gastando sus recursos y su esfuerzos en crear *hashcash proofs-of-work*.

Muchos algoritmos criptográficos hashcash usan una función hash como constructora de bloques, del mismo modo que HMAC o RSA firmas son definidas en una función hash. Hashcash puede ser instanciada con diferentes funciones como hashcash-SHA1 (original), hashcash-SHA256^2 (bitcoin) o hashcash-Scrypt(iter= 1) (litecoin).

El bitcoin usa dos iteraciones hash (denominadas SHA256^2 o *SHA256 function squared*). La causa de esto reside en los ataques parciales realizados al más pequeño, pero estrechamente relacionado SHA-1. Mientras que el algoritmo hashcash se basa en la resistencia a la pre-imagen y, por lo tanto, no es tan vulnerable a los ataques del cumpleaños, un método genérico de blindar SHA-1 frente a estos ataques es realizar una segunda iteración. No se ha realizado un ataque comparable al SHA-256 pero, dado que el diseño del SHA-256 es muy parecido al de SHA-1, es aconsejable la segunda iteración para aplicaciones defensivas.