Enrique Andrés Bolaños Reyes

Universidad Francisco Marroquín

Estructuras de Datos

Carné: 20180521

# Hashing and Collisions

## SHA-3

SHA-3 (Secure Hash Algorithm 3) es el miembro más reciente de la familia de estándares Secure Hash Algorithm, publicado por el NIST el 5 de agosto de 2015. Aunque es parte de la misma serie de estándares, SHA-3 es internamente diferente de la estructura tipo MD5 de SHA-1 y SHA-2.

SHA-3 es un subconjunto de la familia primitiva criptográfica más amplia Keccak, diseñada por Guido Bertoni, Joan Daemen, Michaël Peeters y Gilles Van Assche, basándose en RadioGatún. Los autores de Keccak han propuesto usos adicionales para la función, no (aún) estandarizados por NIST, incluyendo un cifrado de flujo, un sistema de cifrado autenticado, un esquema de hashing de "árbol" para realizar hashing más rápido en ciertas arquitecturas, y cifrados AEAD Keyak y Ketje.

**Aplicaciones**

1. Ethereum es una plataforma de computación distribuida basada en blockchain, pública y de código abierto que ofrece una funcionalidad de contrato inteligente (scripting). Admite una versión modificada del consenso de Nakamoto a través de transiciones de estado basadas en transacciones.

**Ventajas**

1. El algoritmo de hash seguro versión 3 corrige fallas en el cifrado SHA-2 ahora estándar
2. SHA-3 es más lento solo en software. En hardware, supera fácilmente a SHA-1 y SHA-2. Las rutinas criptográficas están siendo manejadas cada vez más por componentes de hardware, y se espera que aumente en el futuro.

**Desventajas**

1. Casi ningún software o hardware del mundo lo admite.
2. Por ahora se debe escribir su propio código y firmware para cada dispositivo que se posea o use.
3. Muchos consideraron que el SHA-3 era mucho más lento que el SHA-2

```
SHAKE128("The quick brown fox jumps over the lazy dog", 256)
f4202e3c5852f9182a0430fd8144f0a74b95e7417ecae17db0f8cfeed0e3e66e
SHAKE128("The quick brown fox jumps over the lazy dof", 256)
853f4538be0db9621a6cea659a06c1107b1f83f02b13d18297bd39d7411cf10c
```

## MD5

El algoritmo de resumen de mensaje MD5 es una función hash ampliamente utilizada que produce un valor hash de 128 bits. Aunque el MD5 fue diseñado inicialmente para ser utilizado como una función criptográfica hash, se ha encontrado que sufre de vulnerabilidades extensas. Todavía se puede usar como una suma de comprobación para verificar la integridad de los datos, pero solo contra la corrupción involuntaria. Sigue siendo adecuado para otros fines no criptográficos, por ejemplo, para determinar la partición para una clave particular en una base de datos particionada.

**Aplicaciones**

1. Los compendios MD5 se han utilizado ampliamente en el mundo del software para proporcionar cierta seguridad de que un archivo transferido ha llegado intacto.
2. Los servidores de archivos a menudo proporcionan una suma de comprobación MD5 (conocida como md5sum) precalculada para los archivos, de modo que un usuario pueda comparar la suma de comprobación del archivo descargado con ella.
3. La mayoría de los sistemas operativos basados en Unix incluyen utilidades de suma MD5 en sus paquetes de distribución

**Ventajas**

1. Es mucho más rápido calcular el resumen de 128 bits para md5 que el resumen de 160 bits para SHA-1
2. El hash MD5 también se utiliza para verificar la autenticidad de un dato sin tener que cifrar realmente los datos

**Desventajas**

1. Un requisito básico de cualquier función criptográfica de hash es que no debería ser computacionalmente viable encontrar dos mensajes distintos que tengan el mismo valor. MD5 no cumple este requisito catastróficamente; dichas colisiones se pueden encontrar en segundos en una computadora doméstica común.
2. Las debilidades de MD5 han sido explotadas en el campo, más infame por el malware Flame en 2012. El CMU Software Engineering Institute considera que MD5 es esencialmente "criptográficamente roto e inadecuado para su uso posterior".

```
3. MD5("The quick brown fox jumps over the lazy dog.") =
4. e4d909c290d0fb1ca068ffaddf22cbd0
```

## Pearson hashing

Pearson hashing es una función de hash diseñada para una ejecución rápida en procesadores con registros de 8 bits. Dada una entrada que consta de cualquier número de bytes, produce como salida un solo byte que depende en gran medida de cada byte de la entrada. Su implementación requiere solo unas pocas instrucciones, más una tabla de búsqueda de 256 bytes que contiene una permutación de los valores de 0 a 255.

**Aplicaciones**

1. Es útil para implementar tablas hash o como un código de verificación de integridad de datos.
2. Se puede hacer para generar hashes de cualquier longitud deseada mediante la concatenación de una secuencia de valores de hash de 8 bits

**Ventajas**

1. Es extremadamente simple.
2. Se ejecuta rápidamente en procesadores de recursos limitados.
3. No existe una clase simple de entradas para las cuales las colisiones (salidas idénticas) sean especialmente probables.
4. Dado un pequeño conjunto privilegiado de entradas (por ejemplo, palabras reservadas para un compilador), la tabla de permutación se puede ajustar de modo que esas entradas produzcan valores hash distintos, produciendo lo que se llama una función hash perfecta.
5. Dos cadenas de entrada que difieren exactamente por un carácter nunca chocan. Por ejemplo, la aplicación del algoritmo en las cadenas ABC y AEC nunca producirá el mismo valor.

**Desventajas**

1. Uno de sus inconvenientes en comparación con otros algoritmos de hash diseñados para procesadores de 8 bits es la sugerida tabla de búsqueda de 256 bytes, que puede ser prohibitivamente grande para un pequeño microcontrolador con un tamaño de memoria de programa del orden de cientos de bytes.

```
h := 0
for each c in C loop
  h := T[ h xor c ]
end loop
return h
```

## BLAKE

BLAKE y BLAKE2 son funciones hash criptográficas basadas en el cifrado de la secuencia ChaCha de Dan Bernstein, pero se agrega una copia permutada del bloque de entrada, XORed con algunas constantes redondas, antes de cada ronda ChaCha. Al igual que SHA-2, hay dos variantes que se diferencian en el tamaño de la palabra. ChaCha opera en una matriz de palabras 4 × 4. BLAKE combina repetidamente un valor hash de 8 palabras con 16 palabras de mensaje, truncando el resultado de ChaCha para obtener el siguiente valor hash. BLAKE-256 y BLAKE-224 usan palabras de 32 bits y producen tamaños de resumen de 256 bits y 224 bits, respectivamente, mientras que BLAKE-512 y BLAKE-384 usan palabras de 64 bits y producen tamaños de resumen de 512 bits y 384 bits, respectivamente.

### Aplicaciones

1. GNU Core Utilities implementa BLAKE2 en su comando b2sum
2. Ruido (protocolo criptográfico), que se usa en WhatsApp y WireGuard incluye BLAKE2 como opción
3. La versión 5 del formato de archivo de archivos RAR admite una suma de control BLAKE2sp de 256 bits opcional en lugar del CRC32 de 32 bits predeterminado; Se implementó en WinRAR v5 +.
4. IPFS permite el uso de BLAKE2b para el hashing de árboles

### Ventajas

1. BLAKE2b es más rápido que SHA-3, SHA-2, SHA-1 y MD5 en arquitecturas de 64 bits x64 y ARM
2. BLAKE2 proporciona una seguridad superior a SHA-2 y similar a la de SHA-3: inmunidad a la extensión de longitud, indiferenciabilidad de un oráculo aleatorio, etc.

### Desventajas

1. No debería ser usado para encriptar contraseñas
2. Existen métodos más eficientes a la hora de hacer implementaciones ASIC

```
BLAKE2b-512("The quick brown fox jumps over the lazy dog")
 = A8ADD4BDDDFD93E4877D2746E62817B116364A1FA7BC148D95090BC7333B3673
   F82401CF7AA2E4CB1ECD90296E3F14CB5413F8ED77BE73045B13914CDCD6A918
```

## SipHash

SipHash es una familia basada en add-rotate – xor (ARX) de funciones pseudoaleatorias creadas por Jean-Philippe Aumasson y Daniel J. Bernstein en 2012, en respuesta a una oleada de "inundación de hash". Ataques de servicio a finales de 2011.

Aunque está diseñado para ser utilizado como una función hash en el sentido informático, SipHash es fundamentalmente diferente de las funciones hash criptográficas como SHA, ya que solo es adecuado como código de autenticación de mensajes: una función hash codificada como HMAC. Es decir, SHA está diseñado para que a un atacante le resulte difícil encontrar dos mensajes X e Y, de manera que SHA (X) = SHA (Y), aunque cualquiera puede calcular SHA (X). SipHash en cambio garantiza que, habiendo visto Xi y SipHash (Xi, k), un atacante que no conoce la clave k no puede encontrar (ninguna información acerca de) k o SipHash (Y, k) para cualquier mensaje Y ∉ {Xi} que no se haya visto antes.

**Aplicaciones**

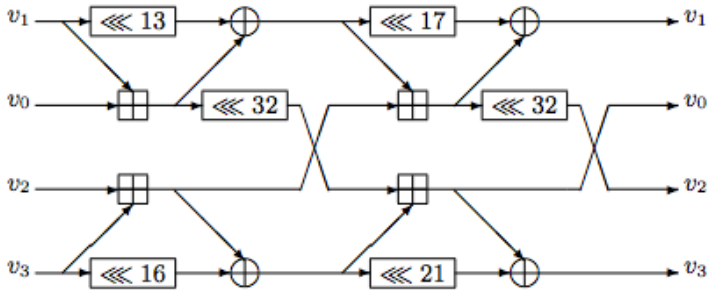
1. Perl 5 (disponible como opción de compilación)
2. Hash Table de Python a partir de la versión 3.4
3. El paquete de software systemd proporciona bloques de construcción fundamentales para un sistema operativo Linux.
4. OpenDNS
5. Hash Table de Haskell

**Ventajas**

1. SipHash calcula el código de autenticación de mensajes de 64 bits o 128 bits a partir de un mensaje de longitud variable y una clave secreta de 128 bits.
2. Rendimiento comparable a las funciones hash no criptográficas, como CityHash, por lo tanto, puede usarse para prevenir ataques de denegación de servicio contra tablas hash ("hash flooding"), o para autenticar la red los paquetes

**Desventajas**

1. Una desventaja es que cada bloque de mensajes debe conservarse mientras el estado está siendo procesado



## Fuentes


https://en.wikipedia.org/wiki/List_of_hash_functions

https://en.wikipedia.org/wiki/SHA-3

Morawiecki, Paweł; Pieprzyk, Josef; Srebrny, Marian (2013). Moriai, S (ed.). "Rotational Cryptanalysis of Round-Reduced Keccak" (PDF). Fast Software Encryption Lecture Notes in Computer Science. 8424. doi:10.1007/978-3-662-43933-3_13. Archived (PDF) from the original on 2013-01-08.

https://www.csoonline.com/article/3256088/why-arent-we-using-sha3.html

https://en.wikipedia.org/wiki/MD5#Applications

https://www.geeksforgeeks.org/applications-of-hashing/

https://131002.net/siphash/siphash.pdf