

Hash

Una función hash criptografía, también conocida como función digesto o función resumen, es un algoritmo matemático que transforma un conjunto de datos de entrada en una salida de longitud fija. Esto nos puede llevar a conseguir que una entrada de gran tamaño tenga una salida de tamaño reducido.

Ejemplos de funciones hash criptográficas son: MD5, SHA-1, SHA-2(SHA-256,...), SHA-3, BLAKE2, etc..... El tamaño del hash $H=h(M)$ dependerá del algoritmo de hash utilizado. Por ejemplo el MD5 es de 128 bits, el SHA-1 es de 160 bits, el SHA-2 de 256 bits.....

Propiedades de las funciones hash:

- Facilidad de cálculo.
- Propiedad unidireccionalidad (de un sólo sentido): dada la salida no debe ser fácil de encontrar la entrada. Aunque con los ataques de diccionario (fuerza bruta) se puede conseguir obtener.
- Propiedad de compresión.
- Propiedad de difusión.
- Propiedad de no predictibilidad.
- Resistencia a preimagen (o resistencia a primera preimagen).
- Resistencia simple a colisiones (o resistencia a segunda preimagen).
- Resistencia fuerte a colisiones.

Resistencia a preimagen (o resistencia a primera preimagen): Las funciones hash (o funciones de un solo sentido) deben presentar resistencia a preimágenes

- Dado un valor resumen $H = h(M)$ debe de ser computacionalmente imposible encontrar una preimagen M para ese valor H .

No sólo el mensaje M original, sino cualquier otra preimagen M .

En resumen, se habla de resistencia (frente) a primera y segunda preimágenes:

- **Resistencia a primera preimagen (o resistencia a preimagen)**

Será computacionalmente difícil que, conocido H , se encuentre un mensaje M tal que $h(M) = H$.

La resistencia a primera preimagen depende de la longitud n del resumen proporcionado por la función hash

Mediante técnicas de fuerza bruta, de media, se obtendrá una preimagen tras $2^{(n-1)}$ intentos (ej. $2^{(256-1)}$ para SHA-256 = $5,789 \times 10^{76}$)

- **Resistencia a segunda preimagen (resistencia simple y fuerte a colisión)**

Será computacionalmente difícil que, conocido M , se encuentre un M' tal que $h(M) = h(M')$.

Será computacionalmente difícil encontrar un par al azar (M, M') de forma que $h(M) = h(M')$.

Mínimos cambios en una entrada pueden producir máximos cambios en la salida.

Listado de algoritmos de hash

- **N-Hash:** Nippon Telephone and Telegraph, 1990. Resumen de 128 bits.
- **Snefru:** Ralph Merkle, 1990. Resúmenes entre 128 y 256 bits. Ha sido criptoanalizado y es lento.
- **MD4:** Ronald L. Rivest, 1990. Resumen de 128 bits.
- **Haval:** Yuliang Zheng, Josef Pieprzyk y Jennifer Seberry, 1992. Resúmenes hasta 256 bits. Admite 15 configuraciones diferentes.
- **RIPEMD:** Comunidad Europea, RACE, 1992. Resumen de 160 bits.
- **MD5:** Ronald L. Rivest, 1992. Resumen de 128 bits. Mejoras sobre MD2 y MD4 (1990), es más lento pero con mayor nivel de seguridad.
- **SHA-0 (o SHA):** National Security Agency (NSA), 1993. Resumen de 160 bits. Vulnerable y reemplazado por SHA-1.
- **SHA-1:** National Security Agency (NSA), 1994. Similar a MD5 pero con resumen de 160 bits.
- **Tiger:** Ross Anderson, Eli Biham, 1996. Resúmenes hasta 192 bits. Optimizado para máquinas de 64 bits (Alpha).
- **Panama:** John Daemen, Craig Clapp, 1998. Resúmenes de 256 bits. Trabaja en modo función hash o como cifrador de flujo.
- **SHA-2 (o SHA-256):** National Security Agency (NSA), 2001-2004. Resúmenes entre 224 y 512 bits (224, 256, 384, o 512). Mejoras sobre SHA-1.
- **SHA-3 (o Keccak):** Guido Bertoni, Joan Daemen, Michal Peeters y Gilles Van Assche, 2015. Resúmenes arbitrarios estándar (224, 256, 384, o 512). Más robusto que SHA-2.

¿Qué algoritmo de hash debería utilizar y con qué longitud?

Pues a diferencia de como ocurre con los algoritmos de cifra simétricos, por ejemplo, la elección de la longitud del hash viene impuesta (en la mayoría de los escenarios) por la función de hash utilizada.

En la actualidad se recomienda el uso de SHA-2, es decir, SHA-256 o superior (SHA.384 o SHA-512), no se debería usar MD5 o SHA-1. Si lo que queremos es anticiparnos a usos futuros, se podría hacer uso de SHA-3(256,384 o 512 bits) o de BLAKE2.

Comparativa de algunos algoritmos Hash:

MD5, SHA-1, SHA-256, SHA-512, SHA-3...

Función	Bits (hash)	Nº. vueltas	Fortaleza preimagen	Fortaleza real preimagen	Fortaleza a colisión	Fortaleza real colisión
MD5	128	64 (4)	2^{128}	$2^{123,4}$	2^{64}	$2^{24,1}$
SHA-1	160	80	2^{160}	$2^{151,1}$ (57/80)	2^{80}	2^{61}
SHA-256	256	64	2^{256}	$2^{255,5}$ (45/64)	2^{128}	$2^{65,5}$ (31/64)
SHA-512	512	80	2^{512}	$2^{511,5}$ (50/80)	2^{256}	$2^{32,5}$ (24/80)
SHA-3	224-512	24	$2^{224-512}$	(256) 2^{128}	$2^{112-256}$	(256) $2^{85,3}$
			2^n		$2^{n/2}$	SHA-3: $2^{n/3}$ (quantum)

Ejemplo de utilización de hashcat:

Cómo utilizar hashcat con un hash de tipo 6 que se corresponde con sha512 para eso utilizamos la opción -m 1800 y la opción -a 0 le indica que vamos a usar sobre el fichero a crackear fichero.txt con un diccionario de contraseñas que está almacenado en la ruta *usr/share/wordlist/rockyou.txt*

```
hashcat -m 1800 -a 0 fichero.txt /usr/share/wordlists/rockyou.txt
```

Calulador de hash:

<https://md5hashing.net/>