educación IT
CODE YOUR FUTURE

Criptografía y Blockchain

Módulo 3

Funciones de *hash*

Fundamentos de las funciones de *hash*

Una función criptográfica de *hash* es un **algoritmo que convierte un mensaje de cualquier tamaño, en un *array* de bits de tamaño fijo y pequeño**, denominado *resumen del mensaje* o *hash.*

Un buen *hash* debe verificar, lo que se lista en el recuadro de la derecha:

- Ser **determinístico** (el mismo mensaje produce el mismo *hash*).
- Ser **irreversible** (debe ser extremadamente dificultoso o imposible recuperar el mensaje original a partir de su *hash*).
- Debe ser computacionalmente **inviable hallar dos mensajes con el mismo *hash*** (*colisión*).
- **Cualquier cambio en el mensaje**, por mínimo que sea, debe resultar en un **extenso cambio en el *hash*** para que sea imposible relacionar los mensajes con sus *hashes* (*efecto avalancha*).
- Debe ser **muy fácil de calcular**.

Aplicaciones

- Verificación de integridad de datos.
- Base para los códigos de autenticación de mensajes basados en *hash* (HMAC).
- Firmas digitales.
- Protocolos de red (TLS, SSH).
- Verificación de contraseñas.
- Identificador de contenidos (por ejemplo, en sistemas de administración de código fuente, SCM, tales como GIT y Mercurial).
- Blockchain y criptomonedas (los bloques de datos se encadenan mediante hashes criptográficos).
- Pruebas de trabajo (minado de criptomonedas, prevención de ataques de denegación de servicios, *spam*).

Seguridad de las funciones de *hash*

Tipos de ataque

Los principales tipos de ataque posibles son:

Ataques de colisión

Intenta hallar dos mensajes que produzcan el mismo *hash*.

Ataques de preimagen

Intenta hallar un mensaje que produzca un *hash* predefinido.

- Un *ataque de colisión* es mucho más sencillo de implementar que un *ataque de preimagen*, porque la búsqueda del mensaje no se limita a un solo objetivo.
- La base de estos ataques es la ***paradoja del cumpleaños***.

Nivel de seguridad

- El nivel de seguridad de una función *hash* es **la complejidad computacional del ataque de colisión**. Se mide en *bits* de seguridad, y depende del tamaño del *hash*.

  Por ejemplo: para SHA-256, la resistencia contra el ataque de colisión es 128 bits y contra el ataque de preimagen, 256 bits.

Niveles de seguridad recomendados por NIST:

- **112 bits** de seguridad: deberían ser suficientes hasta 2030.
- **128 bits** de seguridad: deberían ser suficientes hasta la próxima revolución tecnológica o matemática.

Paradoja del cumpleaños

*Sea **P** la **probabilidad de que una persona cumpla años el mismo día que yo**:*

*¿Cuál es la cantidad de personas que tiene que haber en una habitación para que **P** sea **igual o mayor al 50%**?*

Como la probabilidad es:

$$P = 1 - (364/365)^n$$

Podemos estimar que, para ***P=0,50***, resulta:

$$n >= 253$$

¿Qué tiene que ver esto con las funciones *hash*? Veamos la explicación en el siguiente slide.

Si pensamos las personas como mensajes y su cumpleaños como *hashes*, podemos reinterpretar esta paradoja como:

"La cantidad de mensajes que debo intentar criptoanalizar por fuerza bruta para encontrar dos hashes iguales".

> Recordar que se considera exitoso un ataque de fuerza bruta cuando la probabilidad de éxito sea al menos 50%.

Es el **ataque de la preimagen**. La cantidad de *hashes* que debo calcular es ***253/365 = 0,69*** o sea, aproximadamente el **70%** de todas las claves disponibles en el espacio de claves del algoritmo.

Paradoja del cumpleaños modificada

*Sea **P** la **probabilidad de que dos personas cumplan años el mismo día**.*

*¿Cuál es la cantidad de personas que tiene que haber en una habitación para que **P** sea **igual o mayor al 50%**?*

Como la probabilidad es para ***n>=2***:

$$P = 1 - \frac{[364x363x...x(365-n+1)]}{365^{n-1}}$$

Podemos estimar que, para ***P=0,50***, resulta:

$$n >= 22$$

Nuevamente, pensando las personas como mensajes y su cumpleaños como *hashes*, podemos reinterpretar esta paradoja como:

"La cantidad de mensajes, cuyos hashes debo calcular para encontrar dos hashes iguales que provengan de dos mensajes distintos".

Es el **ataque de colisión**. La cantidad de *hashes* que debo calcular es **22**, que es algo **menor al 10%** del ataque de la preimagen, en definitiva, aproximadamente el 6% de todas las claves disponibles en el espacio de claves del algoritmo.

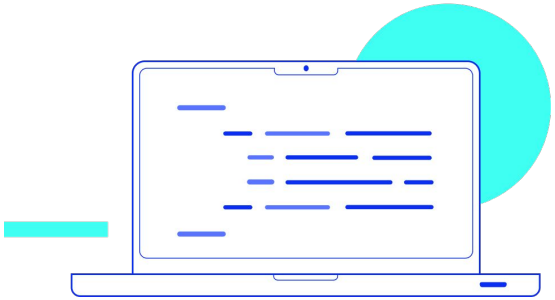
educación IT
CODE YOUR FUTURE

Algoritmos de funciones *hash*

Algoritmos de funciones *hash*

En las próximas slides, veremos en detalle los siguientes algoritmos:

- Familia ***SHA-2***.
- Familia ***SHA-3***.
- Otros algoritmos.



Familia *SHA-2*

Fueron desarrollados por la NSA y publicados por el NIST en 2001. Significa ***Secure Hash Algorithm***. **Se consideran seguros**.

La familia *SHA-2* contiene **los algoritmos más populares**, ***SHA-256*** con un *hash* de 256 bits y nivel de seguridad de 128 bits. También incluye:

- ***SHA-224:*** modificación de *SHA-256* con un IV modificado y una salida truncada a 224 bits. Resistencia a la colisión: 112 bits.
- ***SHA-512:*** similar a *SHA-256* pero para palabras de 64 bits además de palabras de 32 bits. Posee salida de 512 bits y resistencia a la colisión de 256 bits
- ***SHA-384, SHA-512/256 y SHA-512/224:*** modificaciones de *SHA-512* con IV modificado y salida truncada. Los tamaños de *hash* y su correspondiente resistencias a la colisión son 384, 256 y 224 bits y 192,128 y 112 bits, respectivamente.

Familia *SHA-3*

La familia *SHA-3* es la **sucesora de *SHA-2***. Sus algoritmos fueron elegidos a través de una competición, y se basan en diferentes principios que SHA-2. El estándar fue publicado en 2014 y aprobado un año después.

- *SHA-3* no implica que *SHA-2* sea obsoleto, **son algoritmos complementarios**. El algoritmo ganador fue Keccak, desarrollado por criptógrafos belgas.
- **No son tan populares como *SHA-2***. Su uso más notable es la prueba de trabajo de la Blockchain Ethereum.

- Consta de las siguientes **funciones:**
  - *SHA3-224*.
  - *SHA-256*.
  - *SHA3-384*.
  - *SHA-512*.
  - *SHAKE128*.
  - *SHAKE256*.

  Son **similares a sus contrapartes de la familia 2**, tanto en longitud de *hash* como en nivel de seguridad.

- Debido a que ***SHA-3* es más lento**, se crearon ***SHAKE128*** y ***SHAKE256***. Estrictamente hablando, no son funciones *hash* sino *funciones de salida extensible* (***XOF***). Pueden generar resúmenes de tamaño arbitrario y servir de base a las funciones hash. Otra XOF es ***Kangaroo Twelve*** (***K12***).

Otros algoritmos

Funciones hash *SHA-1* y *SHA-0*

SHA-1 fue desarrollado por la NSA y publicado en 1993, pero quedó obsoleto rápidamente. La versión revisada de *SHA-1* fue publicado en 1995 y la anterior de 1993 pasó a denominarse *SHA-0*. En 2017 se halló la primera colisión y *SHA-1* quedó obsoleta. Sin embargo, puede encontrarse aún aplicaciones que la soportan.

Familia MD (*Message Digest*)

Consiste de *MD2*, *MD4*, *MD5* y *MD6* (*MD1* y *MD3* no existen). *MD5* es la función más famosa, muy utilizada antes de ser reemplazada por *SHA-1*.

Familia *BLAKE2*

Consiste de varias funciones, las más conocidas ***BLAKE2s*** y ***BLAKE2b***. La primera produce un hash de 256 bits, la segunda un hash de 512 bits; mantienen su nivel de seguridad en 128 y 256 bits respectivamente. Tienen **mayor margen de seguridad que *SHA-2***, y se asemeja más a *SHA-3*. Son famosos por su **rapidez**.

Su sucesor, ***BLAKE3***, liberada en 2020, soporta resúmenes de tamaño arbitrario igual o mayor a 256 bits. No existe consenso definido aún sobre su nivel de seguridad.

GOST (*GOvernment STandards*)

GOST94 es un *hash* ruso desclasificado en 1994. Produce un hash de 256 bits con nivel de seguridad de 128 bits. Su sucesor es *Streeborg*, publicado en 2012, también conocido como ***GOST2012*** o ***GOST12***. Estándar en **Rusia**.

SM3

Es un *hash* chino publicado en 2010, similar a *SHA-256* en términos de estructura y seguridad. Su nivel de seguridad es 128 bits. Estándar en **China** desde 2016.

Whirlpool

Basado en AES, genera un hash de 512 bits y mantiene su seguridad en 256 bits. No es rápido. Publicado en 2010, no es muy popular. Se usa en *Veracrypt*.

RIPEMD-160

Es el más popular de la familia *RIPEMD*. Genera un hash de 160 bits y un nivel de seguridad insuficiente de 80 bits.

MDC-2 (Código de detección de modificaciones)

Con 128 bits de *hash* y 64 bits de seguridad no se recomienda su uso.

¿Qué algoritmo utilizar?

Requerimiento	Algoritmo
Ningún requerimiento especial.	***SHA3-256***.
Necesitas más **compatibilidad** e interoperabilidad con otro *software*.	***SHA-256*** de la familia *SHA-2*. Pero debe estar preparado para migrar a *SHA-3* ante la posibilidad de un gran salto tecnológico, como la computación cuántica.
Necesitas más **velocidad y seguridad**, y la interoperabilidad no es un problema.	***BLAKE2b*** con *hash* de 512 bits.

¡Sigamos trabajando!