Criptografía - Ingeniería en Computación
Curso 2019

Funciones de Hash

Motivación

¿Qué queremos que cumpla una función de hash en criptografía?

Motivación

¿Qué queremos que cumpla una función de hash en criptografía?

- Dados dos mensajes distintos, para un adversario debería ser computacionalmente difícil generar el mismo valor de hash.

Dentro del área de la computación ¿en qué las podemos utilizar?

Motivación

¿Qué queremos que cumpla una función de hash en criptografía?

- Dados dos mensajes distintos, para un adversario debería ser computacionalmente difícil generar el mismo valor de hash.

Dentro del área de la computación ¿en qué las podemos utilizar?

- Si queremos chequear que dos bases de datos son iguales (backup).

- Si queremos verificar la integridad de los datos, por ejemplo, archivos a descargar desde un sitio web.

- Firma digital

Definiciones - Función de Hash

Función de Hash (informal):

Una función de hash es una función h que tiene, cómo mínimo, las siguientes dos propiedades:

- Compresión: h mapea una entrada x con un largo arbitrario de bits, a una salida $h(x)$ con un largo fijo de bits.

- Facilidad de cálculo: dada la función h y la entrada x, $h(x)$ es sencilla de calcular.

Supondremos:

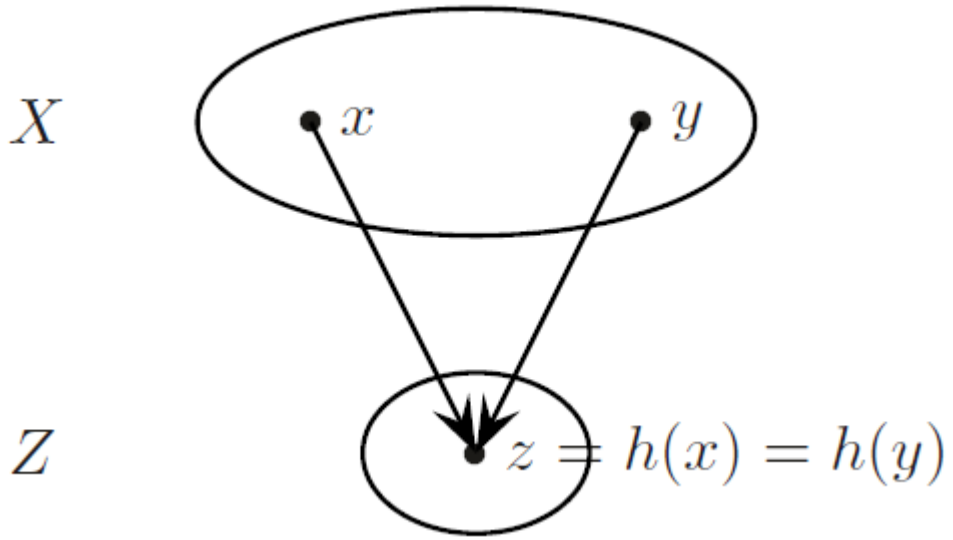
- $\#\, X \geq 2.\#\, Z$

- h es computable en tiempo polinomial

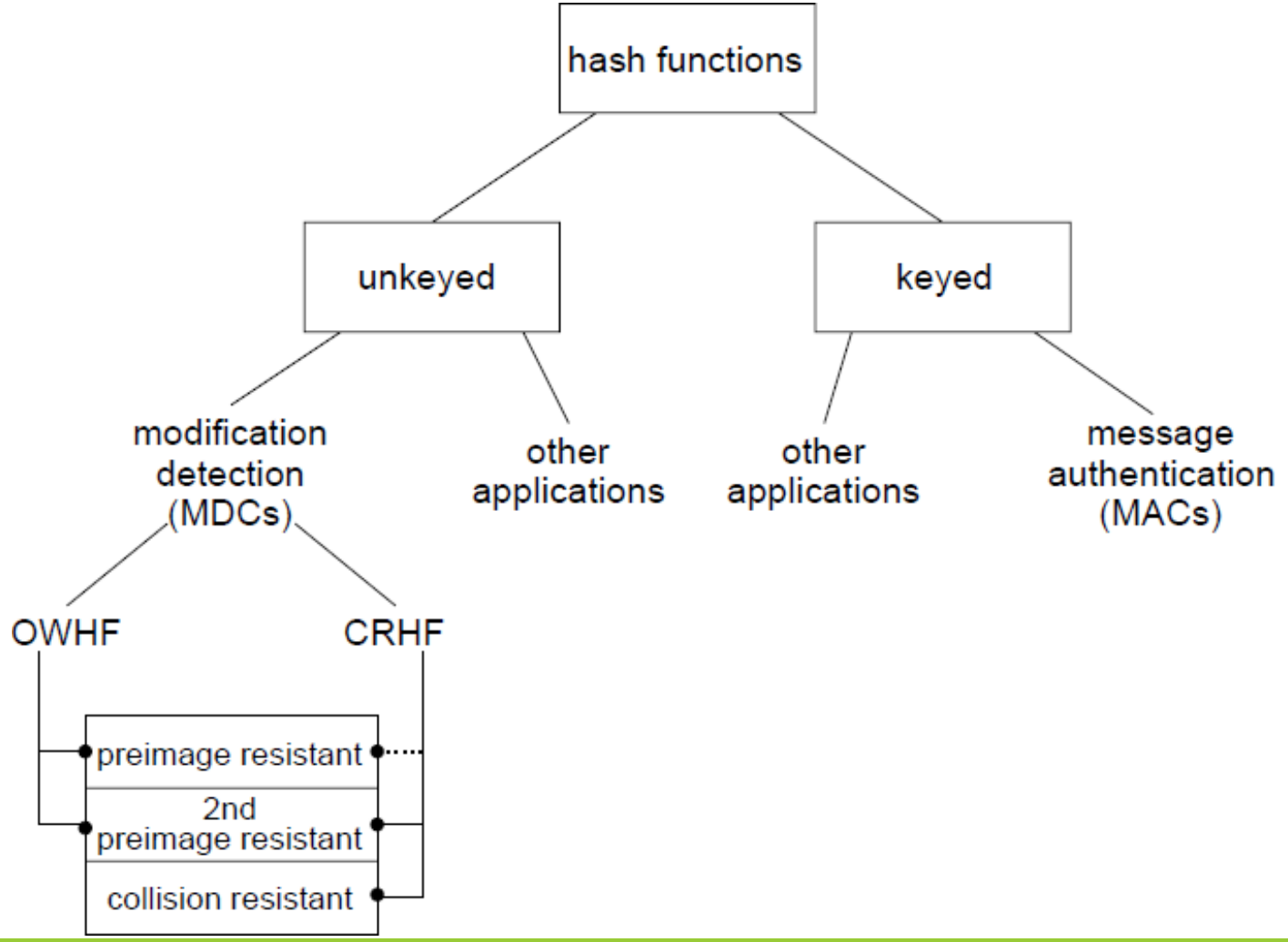
Colisiones

Colisión:

Sea $h: X \to Z$ una función entre dos conjuntos finitos X y Z. Si $x \neq y$ son dos mensajes en X con $h(x) = h(y)$ entonces x e y colisionan, es decir, (x, y) es una colisión.



Clasificación de Funciones de Hash



Propiedades deseables

Para una función de hash h (sin clave) con entradas x,x' y salidas y,y'.

- Diremos que h es resistente a la *primera preimagen* si dado y, es computacionalmente inviable encontrar x tal que $h(x) = y$.

- Diremos que h es resistente a la *segunda preimagen* si dado x, es computacionalmente inviable encontrar x', con $x \neq x'$, tal que $h(x) = h(x')$

- Diremos que h es *resistente a colisione*s si es computacionalmente inviable encontrar x y x', con $x \neq x'$, tal que $h(x) = h(x')$. Notar que hay libertad en la elección tanto de x como de x'.

Propiedades deseables – Ejemplos Menezes

Hash properties required → Integrity application ↓	Preimage resistant	2nd-preimage	Collision resistant	Details
MDC + asymmetric signature	yes	yes	yes†	page 324
MDC + authentic channel		yes	yes†	page 364
MDC + symmetric encryption				page 365
hash for one-way password file	yes			page 389
MAC (key unknown to attacker)	yes	yes	yes†	page 326
MAC (key known to attacker)		yes‡		page 325

Table 9.1: *Resistance properties required for specified data integrity applications.*
†Resistance required if attacker is able to mount a chosen message attack.
‡Resistance required in rare case of multi-cast authentication (see page 378).

Algoritmos

Consideraremos 3 tipos de adversarios A contra la función de hash h

i. Un *collision finder* A, no tiene ningún parámetro como entrada y como salida tiene una colisión (x, y) o sino "fallo"
ii. Un *second preimage finder* A, tiene como entrada el parámetro x y como salida tiene un x' tal que $h(x) = h(x')$ o sino "fallo"
iii. Un *inverter* A, tiene como entrada y y como salida x, tal que $h(x) = y$ o sino "fallo"

En cualquiera de estas situaciones, definimos la probabilidad σ_{A} de éxito de A como:

- $\sigma_{\mathrm{A}} = \mathrm{prob}\{A\ no\ retorne\ \mathrm{fallo}\}$

Denotaremos $T(A)$ como el costo de un algoritmo A, el cual será la suma del tiempo esperado de ejecución más el costo de encodear el string

La constante c será:

- el tiempo en calcular $x \xleftarrow{\boxdot} X$
- el tiempo en calcular $h(x)$ para cualquier $x \in X$
- el tiempo en verificar igualdades

Teorema

Para una función de hash h y $\epsilon > 0$, denotaremos $coll_\varepsilon$, sec_ε e inv_ε al mínimo valor de $T(A)$ sobre los algoritmos definidos en i), ii) y iii), con $\sigma_A > \varepsilon$

Teorema:

Para cualquier función de hash $h: X \to Z$ y $\varepsilon > 0$ se cumple:

i. $coll_\varepsilon \leq sec_\varepsilon + c$

ii. $sec_\delta \leq inv_\varepsilon + 3c$, con $\delta = \epsilon.\left(1 - \frac{\#Z}{\#X}\right) - 2\frac{\#Z}{\#X}$

Corolario

Definición:

Sea $h = (h_n)_{n \in \mathbb{N}}$ una familia de funciones de hash y el tiempo en evaluar h_n es polinomial en n. Llamamos a h resistente a colisiones, resistente a segunda preimagen y resistente al inverso (one-way) si probilísticamente para todo tiempo polinomial i), ii) y iii) la probabilidad σ_A es despreciable

Corolario:

Para una familia $h = (h_n)_{n \in \mathbb{N}}$ de funciones de hash con $\frac{\#Z}{\#X}$ muy pequeño en n, se cumple:

h es resistente a colisiones $\Longrightarrow$ h es resistente a segunda preimagen $\Longrightarrow$ h es resistente al inverso

Si usamos Logaritmo Discreto…

Podemos encontrar una función de hash utilizando un grupo cíclico G = <g>

Qué pasa con las colisiones?

Si usamos Logaritmo Discreto…

Podemos encontrar una función de hash utilizando un grupo cíclico G = <g>

Qué pasa con las colisiones?

Si usamos h: $\mathbb{Z}_d \rightarrow G$ con $h(a) = g^a$ para $a \in \mathbb{Z}_d$

Si usamos Logaritmo Discreto…

Podemos encontrar una función de hash utilizando un grupo cíclico G = <g>

Qué pasa con las colisiones?

Si usamos h: $\mathbb{Z}_d \rightarrow G$ con $h(a) = g^a$ para $a \in \mathbb{Z}_d$

Claramente es resistente a colisiones, es más, no las hay.

PERO no es una función de hash ya que #Z = #G

Si usamos Logaritmo Discreto…

Podemos encontrar una función de hash utilizando un grupo cíclico G = <g>

Qué pasa con las colisiones?

Si usamos h: $\mathbb{Z}_d x \mathbb{Z}_d \rightarrow G$ con $h(a_1, a_2) = g^{a_1+a_2}$ para $a_1, a_2 \in \mathbb{Z}_d$

Si usamos Logaritmo Discreto…

Podemos encontrar una función de hash utilizando un grupo cíclico G = <g>

Qué pasa con las colisiones?

Si usamos h: $\mathbb{Z}_d x \mathbb{Z}_d \rightarrow G$ con $h(a_1, a_2) = g^{a_1+a_2}$ para $a_1, a_2 \in \mathbb{Z}_d$

Ahora es sencillo encontrar colisiones….

$h(a_1 + i, a_2 - i) = h(a_1, a_2)$ para todo $i \in \mathbb{Z}_d$

Si usamos Logaritmo Discreto…

Podemos encontrar una función de hash utilizando un grupo cíclico G = <g>

Qué pasa con las colisiones?

Posible solución….

Utilizamos otro generador más, G= <z>

Si usamos h_z: $\mathbb{Z}_d x \mathbb{Z}_d \rightarrow G$ con $h(a,b) = g^a z^b$ para $a \in \mathbb{Z}_d$

Lo que se viene…

- Hashing long messages
- Timestamps
- MD4/MD5, SHA{0,..,4}
- Ejemplos de colisiones

Muchas gracias!