Pontificia Universidad Católica de Chile
Departamento de Ciencias de la Computación
IIC3253 – Criptografía y Seguridad Computacional
Profesor Marcelo Arenas - Martín Ugarte (Sección 1)
Primer Semestre del 2025

Ayudantía 06 🐷

Funciones de Hash

Ayudante: Diego Rodríguez Cid – darodriguez6@uc.cl

Repaso de conceptos

Funciones de hash

Una función de hash criptográfica H toma una entrada de longitud arbitraria y produce una salida de tamaño fijo (*digest*). Debe cumplir tres propiedades clave:

1. **resistencia a preimagen:** dado y, es difícil encontrar x tal que $H(x) = y$
2. **resistencia a segunda preimagen:** dado x, es difícil hallar otro $x' \neq x$ con $H(x') = H(x)$
3. **resistencia a colisiones:** es difícil encontrar $x \neq x'$ tales que $H(x) = H(x')$

Construcción de Davies–Meyer

Partimos de un esquema criptográfico

$$(Gen, Enc, Dec) \quad \text{sobre} \quad M = K = C = \{0,1\}^*.$$

Para un parámetro de seguridad n, definimos

$$Gen'(1^n) = n,$$

y la función de compresión de bloque fijo

$$h' : \{0,1\}^{2n} \to \{0,1\}^n \quad \text{por} \quad h'(u\|v) = Enc_u(v) \oplus v,$$

En esta construcción:

- El primer bloque $u \in \{0,1\}^n$ se usa como clave de cifrado.
- El segundo bloque $v \in \{0,1\}^n$ es el texto claro.
- Se cifra v con la clave u y luego se aplica XOR con v mismo.

Construcción de Merkle–Damgård

Merkle–Damgård extiende una función de compresión segura a entradas de longitud variable. Se procede así:

1. *Padding*: al mensaje M se le añade un bit '1', ceros y la longitud de M en bits, para que su tamaño sea múltiplo del bloque.
2. *Iteración*: se inicializa con un valor fijo h_0 y se procesa cada bloque l usando la función de compresión: $H_i = (h')^n(m_i \mid\mid H_{i-1})$.
3. *Salida*: el digest final es $h^s(m) := H_l$.

Problema 1. Concepto

¿Por qué el output de una función de Hash debe depender de **todos** los bits de su input? ¿Qué ocurriría en caso contrario?

Solución:
En este escenario, la función queda expuesta a ataques de colisión: un atacante puede alterar la entrada de modo que el hash resultante siga siendo el mismo. Así, sería posible falsificar un mensaje o documento y obtener un valor de hash idéntico al original, dando la impresión de que no ha habido ninguna modificación. Por ejemplo, imaginemos una función de hash que únicamente tiene en cuenta los primeros X bits de la entrada y descarta el resto. Un atacante que conozca esta limitación podría generar una nueva entrada que comparta esos X bits iniciales, pero varíe el resto; ambas producirían el mismo hash, provocando precisamente una colisión.

Problema 2. Resistencia a colisiones

Sean (Gen_1, h_1) y (Gen_2, h_2) dos funciones de hash criptográficas. Se define (Gen, h) como:

$$h^{s_1,s_2}(x) \;=\; h_1^{s_1}(x) \;\|\; h_2^{s_2}(x).$$

Demuestre que si al menos una de las parejas de funciones de hash criptográficas es resistente a colisiones, entonces

$$(\mathsf{Gen}, h)$$

es también resistente a colisiones.

Solución:

Debemos demostrar que la probabilidad de que un adversario gane (es decir, encuentre una colisión) es despreciable. Es decir:

$$\Pr[\text{gane adversario}] \;\leq\; \text{despr}(n),$$

Escrito de otra forma, se tiene:

$$\Pr\big[h(m) = h(m')\big] \;\leq\; \text{despr}(n).$$

Por definición de h, se tiene:

$$\Pr\big[h(m) = h(m')\big] = \Pr\big[h_1(m)\,\|\,h_2(m) = h_1(m')\,\|\,h_2(m')\big] = \Pr\big(h_1(m) = h_1(m') \wedge h_2(m) = h_2(m')\big).$$

Dado que h_1 y h_2 son independientes, se puede separar la probabilidad

$$\Pr\big(h_1(m) = h_1(m') \wedge h_2(m) = h_2(m')\big) = \Pr[h_1(m) = h_1(m')] \;\cdot\; \Pr[h_2(m) = h_2(m')].$$

Sin pérdida de generalidad supongamos que (Gen_1, h_1) es resistente a colisiones (el proceso sería el mismo si la otra lo fuera). Así, se tiene que

$$\Pr[h_1(m) = h_1(m')] \leq \text{despr}(n)$$

Por lo tanto,

$$\Pr[h_1(m) = h_1(m')] \;\cdot\; \Pr[h_2(m) = h_2(m')] \leq \text{despr}(n) \;\cdot\; \Pr[h_2(m) = h_2(m')]$$

Como $\Pr[h_2(m) = h_2(m')]$ es un número menor o igual a 1, se pueded concluir que

$$\Pr[h_1(m) = h_1(m')] \;\cdot\; \Pr[h_2(m) = h_2(m')] \leq \text{despr}(n)$$

Es decir:

$$\Pr\big[h(m) = h(m')\big] \leq \text{despr}(n)$$

En conclusión, (Gen, h) es resistente a colisiones.

Problema 3

Sea Enc una función de encriptación que cumpla con todos los requisitos necesarios de seguridad (esquema criptográfico ideal). Demuestre que h, una versión de Davies–Meyer modificada, NO es resistente a colisiones (en contraste con la construcción de Davies–Meyer original, que sí lo es).

$$h(u \,\|\, v) \;=\; \mathrm{Enc}_u(v) \;\oplus\; u.$$

Solución:

Elegimos arbitrariamente un par (u, v) y luego escogemos otro $u' \neq u$. Definimos

$$v' \;=\; Dec_{u'}\big(Enc_u(v) \oplus u \oplus u'\big).$$

Aplicando $Enc_{u'}$ a ambos lados, y usando que $Enc_k(Dec_k(c)) = c$, obtenemos

$$Enc_{u'}(v') = Enc_{u'}\Big[Dec_{u'}\big(Enc_u(v) \oplus u \oplus u'\big)\Big] = Enc_u(v) \oplus u \oplus u'.$$

Ahora, operamos $\oplus\, u'$ a ambos lados:

$$Enc_{u'}(v') \;\oplus\; u' = \big(Enc_u(v) \oplus u \oplus u'\big) \oplus u' = Enc_u(v) \oplus u,$$

donde hemos usado que $y \oplus (x \oplus x) = y \oplus 0^n = y$. Por tanto,

$$h(u', v') \;=\; Enc_{u'}(v') \oplus u' \;=\; Enc_u(v) \oplus u \;=\; h(u, v),$$

es decir, hemos encontrado una colisión. Concluimos que esta versión modificada de Davies–Meyer *no* es resistente a colisiones.

Lo prometido es deuda: Como fue preguntado en la ayudantía del día 23/04, la razón por la cual esto **no** funcionaría con el esquema de Davies-Meyer original, es que al tratar de definir v', estaríamos definiéndolo a partir de sí mismo, quedaría algo como:

$$v' \;=\; Dec_{u'}\big(Enc_u(v) \oplus u \oplus v'\big).$$

Lo cual no es una definición de v', ya que no hay forma de "despejarlo".

Otra forma de ver esto, es que si seguimos el proceso inverso, partiendo de $h(u', v') \;=\; h(u, v)$, no hay forma de despejar v' para poder llegar a una condición que este debe cumplir para que los *hash* sean iguales.

Problema 4

Sea un esquema de hash basado en Merkle–Damgård con bloques de $\ell = 8$ bits y un IV fijo $H_0 = 0$. La función de compresión viene dada por

$$h(H, B) \;=\; \big(H \oplus B\big) + 1 \pmod{256}.$$

Denotamos por $\mathrm{Hash}(M)$ el valor final tras procesar todos los bloques de un mensaje M.

Diseña un procedimiento que, conociendo únicamente $\mathrm{Hash}(M)$ y un nuevo bloque $X \in \{0, \ldots, 255\}$, calcule directamente

$$\mathrm{Hash}(M \,\|\, X)$$

sin necesidad de conocer el contenido completo de M.

Solución:

Dado que la iteración final de Merkle–Damgård consiste en aplicar la función de compresión al último estado encadenado, basta partir de

$$H \;=\; \mathrm{Hash}(M)$$

y calcular directamente

$$\mathrm{Hash}(M \,\|\, X) \;=\; h\big(H,\; X\big) \;=\; \big(H \oplus X\big) + 1 \;\; \text{mód } 256.$$

Por lo tanto, el procedimiento es:

$$\boxed{\mathrm{Hash}(M\|X) \;=\; \big(\mathrm{Hash}(M) \;\oplus\; X\big) + 1 \;\; \text{mód } 256}.$$