DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC3253 - Criptografía y Seguridad Computacional (I/2023)

Ayudantía 5

Ayudantes: Chris Klempau (christian.klempau@uc.cl)

1. Funciones de Hash

1.1. Definiciones

¿Cuáles son las definiciones formales de las siguientes propiedades y definiciones para una función de hash criptográfica?

a. Función de hash criptográfico:

b. Generador de *keys*:

c. Función de hash *keyed*:

d. Función de compresión:

e. Requerimiento principal - resistencia a colisiones:

Solución:

- **Función de hash criptográfico:**

  El par: algoritmo de hash y generador de *keys*

- **Generador de *keys*:**

  input 1^n y retorna k de largo n

- **Función de hash *keyed*:**

  $h_s : \{0,1\}^* \to \{0,1\}^{l(n)}$

- **Función de compresión:**

  Input $l'(n) >$ Output $l(n)$

- **Requerimiento principal: resistencia a colisiones**

  La probabilidad de que adversario encuentre $h_s(x) = h_s(y) \in \mathcal{H}$ es despreciable, si

$x, y \in \mathcal{M}$, $x \neq y$. Lo anterior implica:

→ Resistencia a segunda preimagen:

dado $x \in \mathcal{M}$, encontrar $y \in \mathcal{M}$ tal que $h_s(x) = h_s(y)$

→ Resistencia a preimagen

dado $y \in \mathcal{H}$, encontrar $x \in \mathcal{M}$ tal que $H_k(x) = y$

- **Hash en la práctica:**

unkeyed, o la *key* es fija.

$h : \{0,1\}^* \to \{0,1\}^l$

Colisión se puede encontrar en tiempo constante.

→ Computacionalmente inviable, ni siquiera 1 colisión se ha encontrado Tardaríamos $1{,}36 * 10^{50}$ años para encontrar una colisión

1.2. Dependencia pura del input

Explique 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 ese caso, la función es vulnerable a ataques de colisiones. Es decir, manipular el input de tal manera, que el output de la función de Hash se mantiene igual.

Por lo tanto, podría falsificar un mensaje o documento, y que h genere el mismo Hash, pareciendo que es el mensaje original.

Por ejemplo, imagine una función de Hash que solo utiliza los 8 primeros bits del input e ignora los demás. Si el atacante sabe esto, puede crear un nuevo input con los mismos 8 bits, pero con los demás bits modificados.

De esta manera, el output de la función de Hash será el mismo, existiendo una colisión.

1.3. Concatenación R.C

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 (Gen_1, h_1) ó (Gen_2, h_2) es R.C, entonces (Gen, h) es R.C.

Solución:

Debemos demostrar que P(gane adversario) $\leq despr(n)$

$$P[h(m) = h(m')] \leq despr(n)?$$

Por definición de h:

$$P[h(m) = h(m')] = P[h_1(m)||h_2(m) = h_1(m')||h_2(m')]$$

Partes iniciales son iguales a partes finales:

$$= P[h_1(m) = h_1(m') \cap h_2(m) = h_2(m')]$$

Pero como son funciones de hash distintas, sus probabilidades son independientes:

$$= P[h_1(m) = h_1(m')] \cdot P[h_2(m) = h_2(m')]$$

Ahora, s.p.g, decimos que h_1 es R.C., por lo que su probabilidad es una función despreciable:

$$= despr(n) \cdot P[h_2(m) = h_2(m')]$$

$$= despr(n) \cdot \underbrace{P[h_2(m) = h_2(m')]}_{\leq 1} \leq despr(n)$$

En conclusión, $P[h(m) = h(m')] \leq despr(n)$.

Por lo tanto, Gen, h es R.C.

2. Merkle-Damgård

2.1. Padding

¿Por qué en la construcción de Merkle-Damgård $Pad(m)$ se incluye el largo del mensaje? ¿Qué propiedad necesaria se rompería en caso contrario?

Solución:

Si $|m|$ no es el múltiplo del largo del bloque, entonces:

- $Pad(m) = m||p_m$

Ahora, si hasheamos directamente un m' tal que $m' = m||p_m = Pad(m)$

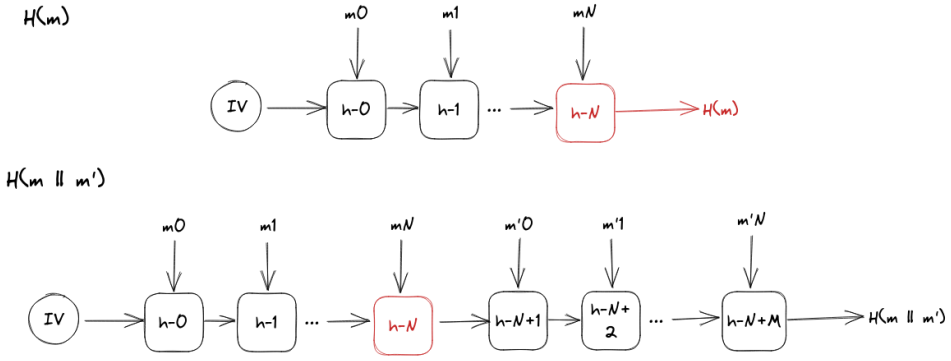
Entonces encontramos una colisión, ya que m' y m tienen el mismo Hash.

2.2. Ataque de extensión de largo

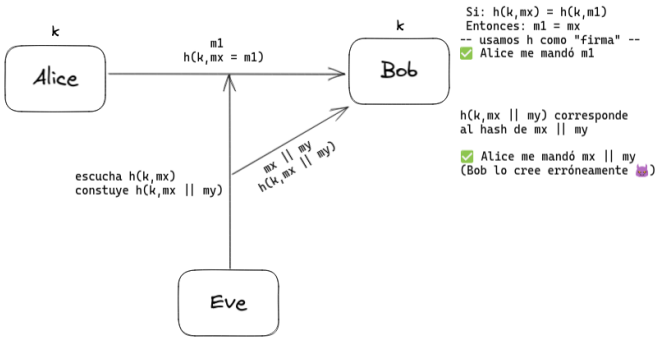
Considere una función de hash $\mathcal{H}$ usando la construcción de Merkle-Damgård.

- Demuestre que un adversario que conoce el hash $H(m)$ de un mensaje m, pero no m en sí, puede generar un mensaje m' y calcular el hash $H(m||m')$.
- ¿Qué implicancia tiene lo anterior en usar Merkle-Damgård para verificar integridad de mensajes?

Solución:



Como se muestra en la imagen, $H(m)$ aparece en $H(m||m')$. Por lo tanto, podemos construir un Hash extendido, para $H(m||m')$ a partir de $H(m)$, sin tener m.



Solo Alice y Bob tienen la llave k. Luego, un *eavesdropper* Eve podría escuchar un mensaje hasheado en un canal inseguro, y construir un nuevo hash de un mensaje malicioso. Luego,

Bob creería incorrectamente que Alice se lo mandó, bajo la premisa que hash funciona como una "firma", y el hash coincide con el mensaje enviado.

Para solucionar el problema necesitamos HMAC! (semana siguiente).

3. Davies-Meyer

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

$$h(u||v) = Enc_u(v) \oplus u$$

Solución:

Elegimos (u, v) arbitrariamente, y luego un u' arbitrariamente, tal que $u \neq u'$

Luego, podemos definir una variable $v' \neq v$ como:

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

Aplicamos $Enc_{u'}(...)$ a ambos lados:

$$Enc_{u'}(v') = Enc_{u'}[Dec_{u'}(\underbrace{Enc_u(v) \oplus u \oplus u'}_{c})]$$

Luego, como $Enc_k(Dec_k(c)) = c$:

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

Aplicamos $(\oplus u')$ a ambos lados:

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

Pero $y \oplus (x \oplus x) = y \oplus (0^n) = y$, entonces se simplifica a:

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

Que es igual a nuestra definición de hash:

$$h(u', v') = h(u, v)$$

Por lo tanto, encontramos una colisión y nuestra definición alternativa de Davies-Meyer NO es resistente a colisiones.