

IIC3253

Repaso: de AES a Funciones de Hash

¿Cómo construimos una función de hash?

Partamos simplificando el problema:

Input de largo arbitrario \rightarrow output de largo fijo

Input de largo fijo \rightarrow output de largo fijo

$$h : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$$

¿Podemos usar un esquema de cifrado?

$Enc_k(m) : \{0, 1\}^n \rightarrow \{0, 1\}^n$ para cada llave $k \in \{0, 1\}^n$

¿Y si vemos la llave como otra variable?

$Enc_k(m) = Enc(k, m) : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$

Toma $2n$ bits y devuelve n bits 

Definimos la función $h'(x) = Enc_{x_0}(x_1)$, donde x_0 son los primeros n bits de x y x_1 los últimos n bits.

¿Es h' resistente a pre-imagen?

¿Dado $y \in \{0, 1\}^n$, puedo encontrar un valor x tal que $y = h'(x)$?

Sea $k \in \{0, 1\}^n$ un valor cualquiera. Definimos x como $k || Dec_k(y)$.

Tenemos $h'(x) = Enc_k(Dec_k(y))$

Demuestre que $Enc_k(Dec_k(y)) = y$

¿Cómo lo arreglamos?

Davies-Meyer

$$h'(x) = h'(x_0 || x_1) = Enc_{x_0}(x_1) \oplus x_1$$

Si Gen viene de un esquema criptográfico *ideal*,
entonces h' es resistente a colisiones

Llamaremos a esta primera función de hash de
largo fijo nuestra *función de compresión*.

¿Cómo construimos una función de hash?

Partamos simplificando el problema:

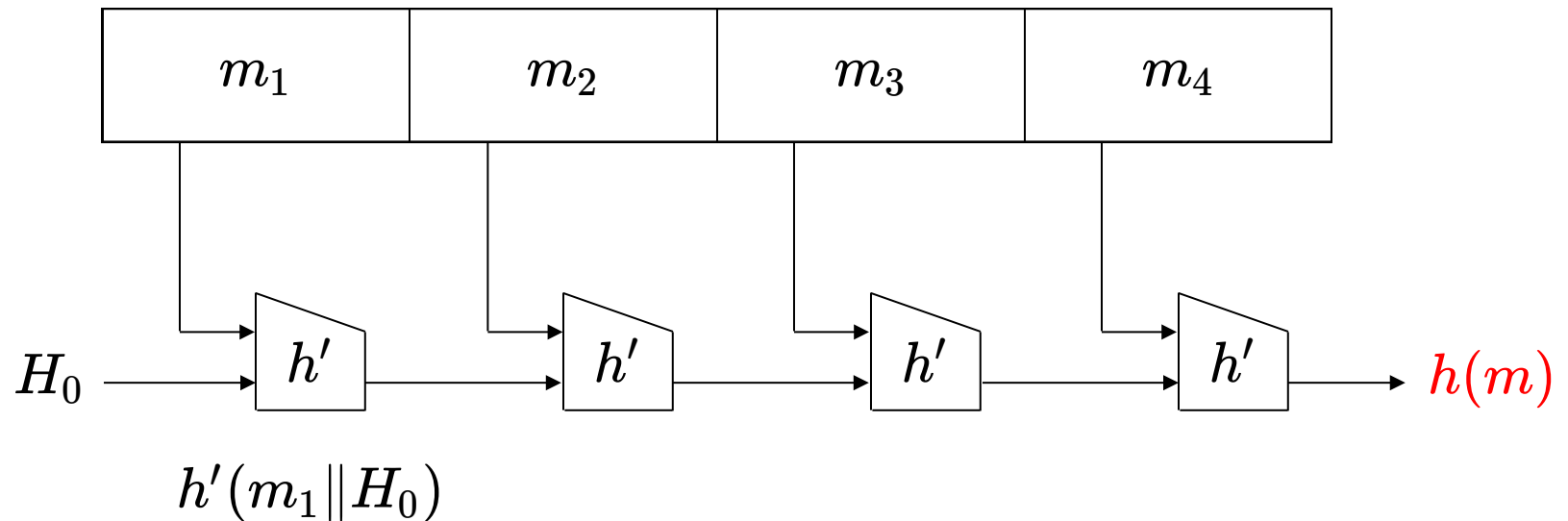
Input de largo fijo

$$h : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$$

¿Input de largo múltiplo de n ?

Merkle-Damgård

bloque
de n bits




¿Resistente a colisiones?

Nos gustaría demostrar que si h' es resistente a colisiones, entonces h es resistente a colisiones

Por contrapositivo: Un algoritmo que encuentra colisiones para h permite encontrar colisiones para h'

¿Teniendo a una colisión en h , puedo encontrar una colisión en h' ?

Supongamos que $h(m_1) = h(m_2)$ con $m_1 \neq m_2$

Si m_1 y m_2 tienen el mismo largo, vamos
a encontrar una colisión en h' 

¿Y si no tienen el mismo largo?

Podríamos **no** tener una colisión en h'
si conociéramos x_0 tal que $h(x_0) = H_0$

Definimos H_0 como un *nothing-up-my-sleeve number*, y
suponemos que nadie conoce una pre-imagen x_0