Clase 4

8 de abril de 2024

Esquema Encripción (Cifrado) → Provee Confidencialidad

→ No detecta alteraciones en mensaje

No provee

INTEGRIDAD/AUTENTICACIÓN

MAC Consigue detectar si el mensaje ha sido alterado (o si es autentico)

Message Authentication Codes

Esquema MAC (Gen, Mac, Vrfy)

- Gen: $k \leftarrow Gen(1^n)$ 1^n -> cadena de longitud d
- Mac: $t \leftarrow Mac_k(m)$ emisor envía <m,t>
- Vrfy: $b := Vrfy_k(m,t)$ b = 1 si es válido

Mac y Vrfy van a tener que pasar la prueba Mac Force (que detecta falsificacion)

Experimentos:
EAV (eavesdropping): no hay oraculo
CPA: hay oraculo, puedo consultar m0 o m1 (buscamos que no sea deterministico)
MAC-F: hay oraculo, puedo consultar todo menos el mensaje que emito

Seguridad de MAC

Experimento $\boldsymbol{MAC - Forge_{A,\pi}(n)}$

1) $k \leftarrow Gen(n)$
2) El adversario A recibe 1^n y acceso al oráculo $Mac_k(.)$
   Sea Q, conjunto de todas las preguntas que hace A al oráculo $Mac_k(.)$
   El adversario A emite un par <m,t>.de igual longitud
3) La salida del experimento es 1 (ÉXITO) sí y sólo sí:
   (1) $Vrfy(m,t) = 1$ y
   (2) $m \notin Q$

MAC es seguro (infalsificable) ante un ataque de mensaje elegido sí y sólo sí:

$$\boldsymbol{Pr[MAC - Forge_{A,\pi}(n) = 1] \leq negl(\quad)}$$

La probabilidad de acertar no va a ser cero, pero pedimos que sea negligible.

MAC no protege contra ataques de REPLAY

- Nº de secuencia
- Timestamps

Los ataques de replay son usar un mensaje con una etiqueta mas de una vez

Se solucionan incorporando Nros de secuencia o timestamps

Construcción de MAC seguros para mensajes de longitud fija n

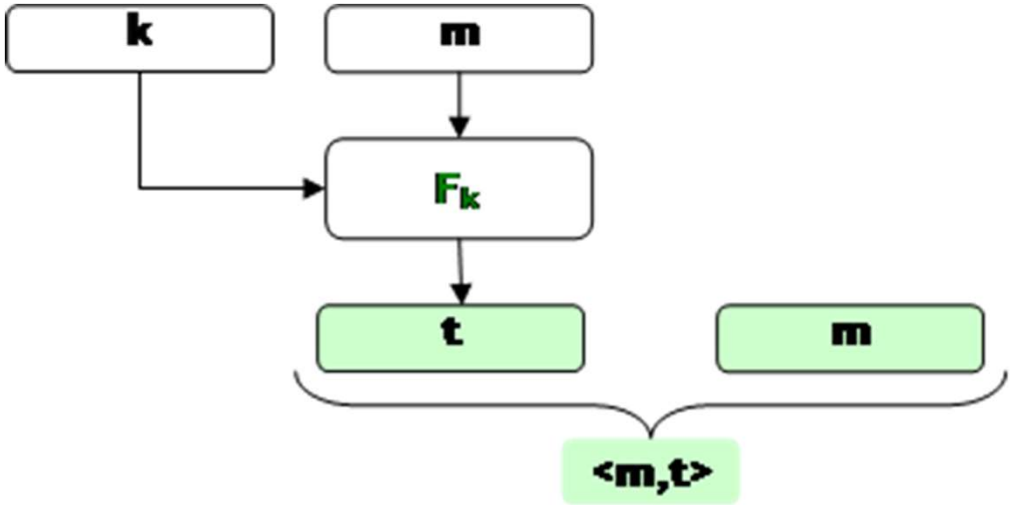
Esquema MAC (Gen, Mac, Vrfy)

- Gen: $k \leftarrow \{0,1\}^n$
- Mac: $t \leftarrow F_k(m)$ $|m|=|k|=n$
- Vrfy: $b := Vrfy_k(m,t)$
  - Si $|m| \neq |k| \Rightarrow 0$
  - Sino, si $F_k(m)==t \Rightarrow 1$

MAC seguro PARA MENSAJES DE LONGITUD FIJA n

F_k Función Seudoaleatoria

Cuando el mensaje es de longitud fija, hacemos tags de la misma longitud del mensaje



Longitud variable: antes teniamos un solo bloque. Ahora tenemos multiples bloques de longitud fija.
Las sugerencias son intentos de solucionar los problemas que surgen, pero son todos incorrectos.

Construcción de MAC seguros para mensajes de longitud variable

Sugerencia 1:

XOR entre los bloques, autenticar el resultado: $t := Mac'_k(\oplus_i m_i)$

y emitir $\langle m_1, m_2, \ldots m_d; t \rangle$

El adversario puede falsificar un tag válido sobre un nuevo mensaje, cambiando el mensaje original como para que el XOR de los bloques no cambie.

Sugerencia 2:

Autenticar cada bloque por separado: $t_i := Mac'_k(m_i)$

y emitir $\langle m_1, m_2, \ldots m_d; t_1, t_2, \ldots, t_d \rangle$

El adversario puede cambiar el orden de los bloques y calcular un tag válido sobre ellos (ej. $m_d, \ldots, m_2, m_1$)

Sugerencia 3:

Autenticar cada bloque junto con número de secuencia: $t_i := Mac'_k(i \| m_i)$

y emitir $\langle m_1, m_2, \ldots m_d; t_1, t_2, \ldots, t_d \rangle$ i = num de secuencia

El adversario puede mezclar bloques de diferentes mensajes

Ej: de $\langle m_1, m_2, \ldots m_d; t_1, t_2, \ldots, t_d \rangle$ y $\langle m'_1, m'_2, \ldots m'_d; t'_1, t'_2, \ldots, t'_d \rangle$ es válido $\langle m_1, m'_2, m_3, m'_4 \ldots t_1, t'_2, t_3, t'_4 \ldots \rangle$

A partir de MAC (Gen', Mac', Vrfy') para mensajes de longitud n.

- Gen: Gen'
- Mac: $|m| = L < 2^{n/4}$ $|k| = n$

El mensaje m se parte en d bloques de longitud n/4 (se completa con ceros)

elegir $r \leftarrow \{0,1\}^{n/4}$ r = random de longitud n/4

$t_1 = Mac'_k(r \| L \| 1 \| m_1)$

$t_2 = Mac'_k(r \| L \| 2 \| m_2)$

...

$t_d = Mac'_k(r \| L \| d \| m_d)$

$\Rightarrow$ emite $\langle r, t_1, t_2, \ldots, t_d \rangle$

Sugerencia 1: hacer un XOR sobre todos los m_i (todos bloques de igual longitud). Esto genera un bloque de igual longitud que el resto. Lo que hago es mandar ese bloque como mensaje. Es inseguro porque si intercambio el primer bit del bloque uno con el primer bit del bloque dos, el XOR me da igual.

CBC-MAC

Esta solucion es ineficiente. Se parte el mensaje en 4 partes, generando d bloques.
El mensaje termina aumentando en 4 su tamaño.
La solucion es CBC-MAC.

Los tags que se generan se van usando para el siguiente bloque.

CBC-MAC

Al ser todos 0, no cambia m1 con el XOR. Esto permite reutilizar la funcion previa de CBC que usaba el IV

La ultima etiqueta (tn) es la unica que sirve. Lo que logro es que el atacante no pueda mover el orden de los mensajes, o el orden de los bits dentro de cada mensaje. Entonces, lo que se termina enviando es el mensaje completo y tn, que son de la misma longitud.

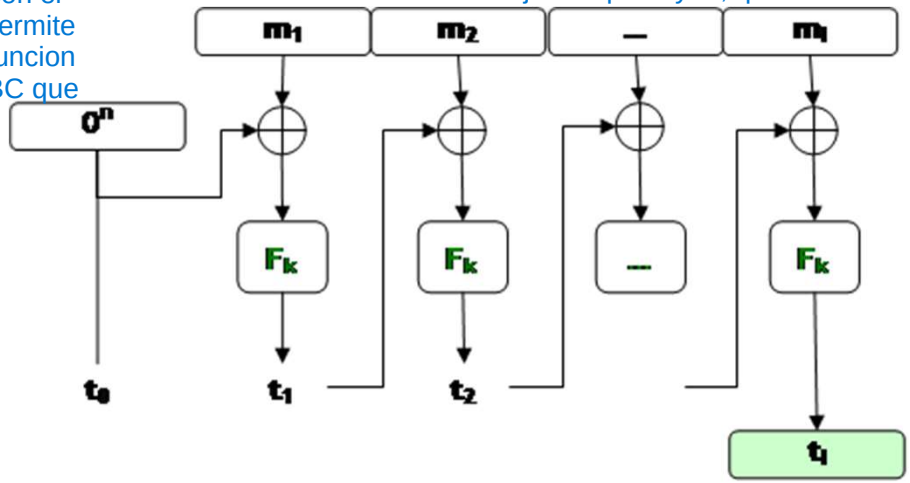
CBC-MAC (Gen, Mac, Vrfy)

- Gen: $k \leftarrow \{0,1\}^n$
- Mac:
  - $|k| = n$
  - $|m| = L(n).n$

El mensaje m se parte en l bloques de longitud n

$t_0 = 0^n$

$t_i = F_k(t_{i-1} \oplus m_i) \Rightarrow$ emite t_l



La construcción anterior es infalsificable SÓLO si se permiten mensajes de una misma longitud.

Opciones seguras para CBC-MAC de distintas longitudes:

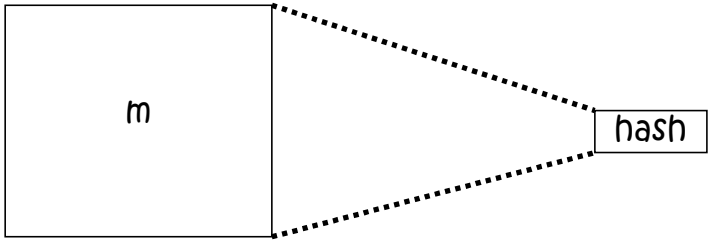
Opción 1: $k_l := F_k(|m|)$ y $t \leftarrow CBC - MAC_{kl}(m)$

Opción 2: $m' := |m| \,\|\, m$ y $t \leftarrow CBC - MAC_k(m')$

Opción 3:

$$k_1 \leftarrow \{0,1\}^n$$
$$k_2 \leftarrow \{0,1\}^n$$
$$t \leftarrow CBC - MAC_{k1}(m)$$
$$\hat{t} \leftarrow F_{k2}(t)$$

Función de HASH



comprime la entrada a una longitud fija.

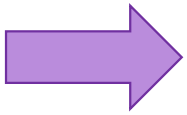
Dado que la longitud del hash es mas corta que la del mensaje, hay que tener cuidado con las colisiones.

Permiten construir Esquemas Mac seguros.

Colisiones

Para $x \neq x'$ resulta $H(x) = H(x')$

Las colisiones van a existir



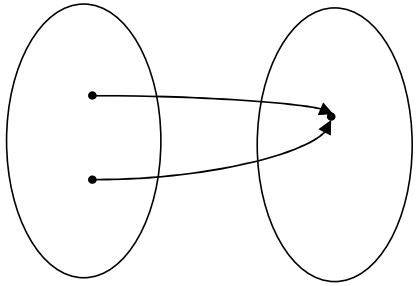
Nociones de seguridad más débiles:

- Resistente a segundas preimágenes
- Resistente a preimagen

Niveles de Seguridad

1. Resistente a Colisiones

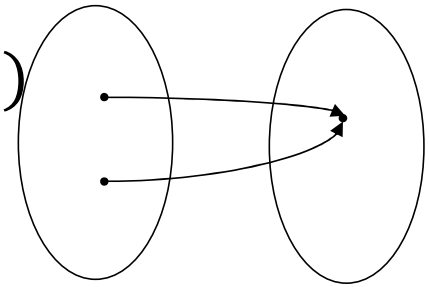
Encontrar $\boldsymbol{x \neq x'}$ tal que $\boldsymbol{H^s(x) = H^s(x')}$



2. Resistente a Segundas Preimágenes

Dado x, encontrar $\boldsymbol{x \neq x'}$ tal que $\boldsymbol{H^s(x) = H^s(x')}$

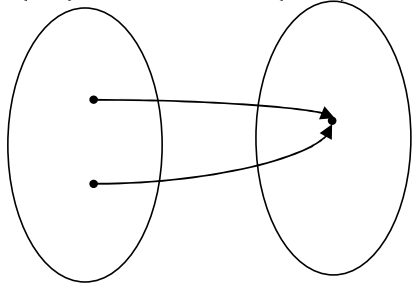
Este no es igual al punto 1, dado que en el primero son dos X cualquiera, y en este parto de un X ya fijado.



3. Resistente a Preimagen

Dado $\boldsymbol{H^s(x) = y}$, encontrar algún $\boldsymbol{x'}$ tal que $\boldsymbol{H^s(x) = H^s(x')}$

En este ataque yo parto de un hash, y quiero buscar un X' que me de ese hash y sea distinto al X original



Tamaño de Salida

Para ser segura, una función de hash resistente a colisiones necesita tener una salida que sea mayor de 160 bits (2^{80} cómputos).

Es una condición necesaria pero no suficiente.

(Porque si el algoritmo es malo no puedo hacer nada)

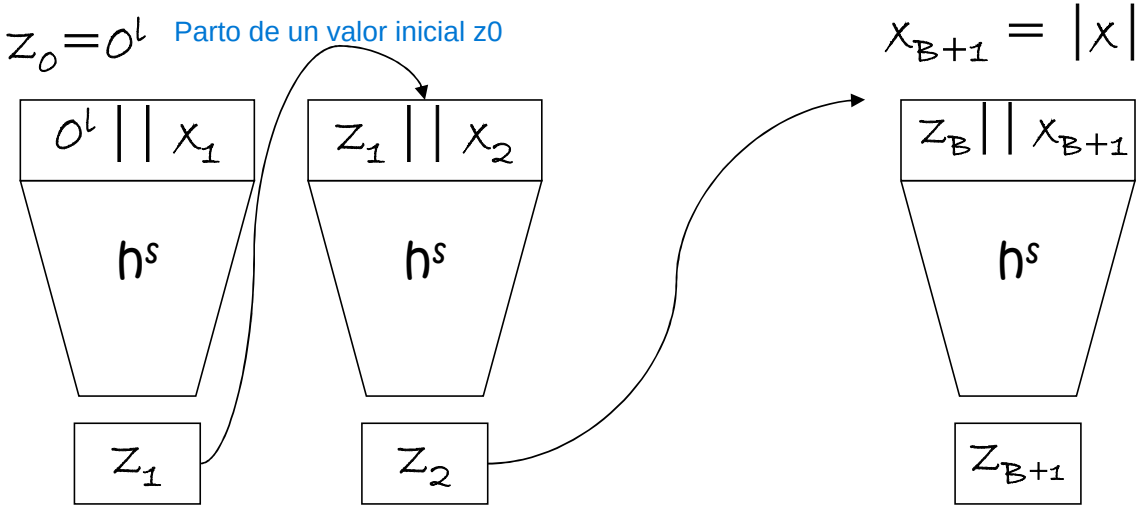
$H^s(x)$ Resistentes a Colisiones en la práctica

Transformación de Merkle Darmgard

para mensajes de longitud variable.

$|x| = L < 2^{l(n)}$ $\quad x = x_1 x_2 x_3 \ldots x_B \quad |x_i| = l$

Se usa: $h^s: \{0,1\}^{2l} \rightarrow \{0,1\}^l$ segura



Genero un hash z1, y lo combino con el mensaje siguiente para lograr algun tipo de encadenamiento

$$z_i = h_s(z_{i-1} \,||\, x_i)$$

Una colisión en H^s sólo puede ocurrir si hay una colisión en h^s.

Si el algoritmo no tiene colisiones, la transformacion tampoco

NMAC (Nested Mac)

- Gen: emite (s, k_1, k_2)

$|k_1| = |k_2| = n$

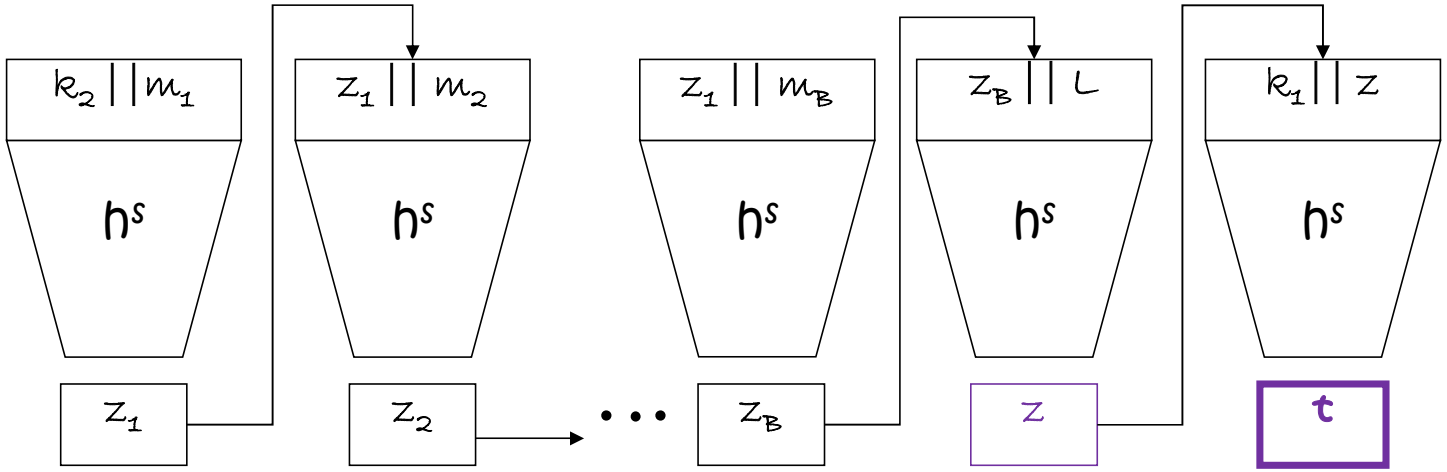
- Mac: emite $<m,t>$

$m = m_1 m_2 m_3 \ldots m_B$

$|m| = L$

$|m_B| = n$

$t := \boldsymbol{h^s_{k_1}(H^s_{k_2}(m))}$



Primero se parte de k2. Luego se hace la transformacion de Merkle. Finalmente se utiliza k1.

$\boldsymbol{H^s_{k_2}(m)} = z$

$\boldsymbol{h^s_{k_1}(z)} = t$

HMAC hash-based mac

- Gen: emite (s, k)

$|k| = n$

- Mac: emite <m,t>

$m = m_1 m_2 m_3 \ldots m_B$

$|m| = L$

$|m_B| = n$

opad = n veces byte "0x36"

ipad = n veces byte "0x5C"

IV = constante fija de longitud n

opad e ipad siempre valen esos valores

$t := H_{IV}^{s}\left((k \oplus opad)||H_{IV}^{s}((k \oplus ipad)||m)\right)$