GUÍA 3: MESSAGE AUTHENTICATION CODES (MAC) Y FUNCIONES DE HASH - SOLUCIONES

Ejercicio 1:

El MAC es seguro si un adversario con acceso a un oráculo Mac, no puede generar un par <m,t> válido (es decir un par mensaje-etiqueta que pueda verificarse) para un mensaje m distinto de los ya consultados al oráculo, excepto con probabilidad despreciable.

Es decir, se considera que el atacante puede calcular etiquetas sobre cualquier mensaje de su elección, y luego calcular un par <m,t> válido sobre un mensaje diferente a los calculados con la ayuda del oráculo.

1) $Mac_k(m) = G(k) \oplus m$, donde $G(x)$ es un generador seudoaleatorio.

No es seguro.

Si el adversario tiene posibilidad de calcular etiquetas sobre el mensaje que quiera, puede efectuar:

$Mac_k(m) = Mac_k(0^n) = G(k) \oplus 0^n = G(k)$

Luego, puede emitir un par <m', t'> válido para algún m' distinto de m= 0^n , ya que conoce $G(k)$

Así, puede emitir <m', t'> con $t' = Mac_k(0^n) \oplus m' = G(k) \oplus m'$

2) $Mac_k(m) = k \oplus first\_k\_bits(m)$

No es seguro.

Ejemplo:

Sea $m \in \{0,1\}^{2n}$, y $k \in \{0,1\}^n$

Entonces $Mac_k(m) = k \oplus first\_n\_bits(m)$

Si el adversario tiene posibilidad de hacer consultas al oráculo, puede efectuar:

$Mac_k(m) = Mac_k(0^{2n}) = k \oplus 0^n = k$ y luego calcular pares <m', t'> válido para algún m' distinto de 0^{2n}, ya que conoce k .

3) $Mac_k(m) = Enc_k(|m|)$ con Enc siendo CPA-Secure.

No es seguro.

Si el adversario emite <m,t>, la validación de t es mediante $Dec_k(t) = |m|$.

Por ejemplo, sea $m_1 = 00000000$, entonces $Mac_k(m_1) = t_1$ y el mismo Mac tendrá $m' = 11111111$, ya que $Dec_k(t') = 8$

Ejercicio 2:

M = el auto es azul y el lapiz rojo

M'= el auto es rojo y el lapiz azul

Ejercicio 3:

a) Si se calcula el hash MD5 de la cadena "hoy es el ultimo lunes de marzo".

```
$echo hoy es el ultimo lunes de marzo>texto.txt
$openssl dgst –md5 texto.txt
MD5(texto.txt)=828416d2688bd0bffb0a20c132ffc6cc
```

b) Si se calcula el hash SHA-1 de la misma cadena:

```
$echo hoy es el ultimo lunes de marzo>texto.txt
$openssl dgst –sha1 texto.txt
SHA1(texto.txt)= 480bfc914b333bfd9517d94a2b8ace694bc28bf5
```

c) ¿Qué diferencias se observan?

SHA1 es de mayor longitud.

Ejercicio 4:

Rta: Hacerlos con echo. (Ej. $echo 7 siete>siete.txt)
acuña: 7 siete
centurión: 6 seis
hernandez: 8 ocho
palacios: 10 diez
rossi: 5 cinco
sanchez: 2 dos
garcía: 1 uno
zubeldía: 10 diez.

Ejercicio 5:

a) h(x)

- toma una entrada de longitud arbitraria y produce una salida de longitud fija
- es fácil de calcular
- No es válida para criptografía, ya que si analizamos los tres niveles de seguridad:

1) Resistencia a preimágenes: no cumple.

Dado $y = H(m)$, para un valor m elegido aleatoriamente, no debería ser factible para un adversario PPT encontrar algún valor x' tal que $H(x') = y = H(m)$.

Dado por ejemplo, y = 0, no es posible decir qué mensaje lo produjo.

Sin embargo, es posible encontrar algún valor x' tal que $H(x') = 0$, por lo tanto no cumple el nivel de resistencia a preimágenes.

2) Segunda imagen: no cumple.

Dado m y H(m), no debería ser factible para un adversario PPT encontrar x' tal que H(x') = H(m)

Dado por ejemplo, m = 0101, y H(0101) = 0^{32}, no es difícil encontrar que, por ejemplo, x'= 0100 tiene H(x') = 0^{32} también.

3) Resistencia a colisión: no cumple.

Encontrar $x \neq x'$ tal que H(x)=H(x') es fácil.

b) Si consideramos que la longitud de la entrada es aleatoria, entonces la salida es aleatoria con un 50% de probabilidad para cada una de las dos salidas posibles. La posibilidad de una colisión es la misma que la de obtener "cara" dos veces seguidas o "ceca" dos veces seguidas, es decir, es de un 50% (P(colision) = 0,5)

Ejercicio 6:

a. Describir la construcción CBC-MAC e indicar para qué sirve.

[Katz Cap. 4.5] La construcción CBC-MAC sirve para generar un código de autenticación para mensajes de longitud variable (longitud mayor que n, siendo n la longitud de la clave)
[Katz Construcción 4.9] Descripción de la construcción básica **CBC-MAC**:

Sea F una **función seudoaleatoria** y se fija l, una función de longitud.
El CBC-MAC básico consiste en:

1) **Gen** = Elige en forma uniforme y aleatoria **k** $\leftarrow \{0,1\}^n$

2) **Mac** = dada una **clave k** y **un mensaje** m, de longitud $l(n) \cdot n$, hacer lo siguiente:

1. dividir m en $m = m_1 m_2 ... m_{l(n)}$, donde cada m_i es de longitud n, y establecer $t_0 := 0^n$

2. Para $i := 1$ hasta $l(n)$, establecer $t_i := F_k(t_{i-1} \oplus m_i)$

   Emitir $t_{l(m)}$ como etiqueta.

3) **Vrfy** = input: **clave k** , **un mensaje** m **,** de longitud $l(n) \cdot n$ **,** y una **etiqueta t**$\in \{0,1\}^n$ **,**

   output: 1 si y sólo si $t \stackrel{?}{=} Mac_k(m)$

La construcción básica CBC-MAC brinda una Código de Autenticación de Mensajes seguro para mensajes de **longitud fija**, con l(n) polinomio.

El esquema básico **no es seguro** en el caso general en el que mensajes de diferentes longitudes requieran ser autenticados.

b. Comparar CBC-MAC con CBC-mode para encripción.

CBC-MAC	CBC.-mode encripción
Usa IV = 0^n (o cualquier IV fijo, no aleatorio) porque un IV aleatorio no sería seguro. Si se hace XOR del primer bloque con un IV aleatorio, cuando el IV cambie basta que se cambien los bits del primer bloque del mensaje en función del cambio de IV para obtener un mensaje alterado con la misma etiqueta. Ej.: Sea $m_0 = 01000000$ y $IV = 10100000$. El primer t1 será el resultado de $F_k(m_0 \oplus IV) = F_k(11100000)$ Luego si se tiene otro IV, por ejemplo $IV = 01011100$, basta con cambiar sólo el primer bloque por $m_0 = 10111100$ para que el resultado t1 sea el mismo del mensaje anterior (aunque se dejen intactos los otros bloques del mensaje) y por lo tanto la etiqueta final también.	usa IV aleatorio y esto es crucial para obtener seguridad (evita ataques de texto plano elegido)
Sólo el bloque final se emite como resultado	Todos los valores intermedios se emiten como resultado.

c. ¿Cuáles son, según Katz, las opciones seguras de uso de CBC-MAC?

La construcción básica de CBC-MAC es una construcción segura sólo cuando la longitud de los mensajes es fija. Si un adversario puede obtener etiquetas MAC para mensajes de distintas longitudes, el esquema deja de ser seguro.

Demostración:

Si se elige un m de longitud n (con $|m| = |k| = n$), entonces se puede pedir el CBC-MAC de dicho m y almacenarlo en el conjunto Q de mensajes para los cuales ya se ha calculado la etiqueta. La etiqueta sería: $t := F_k(m)$.

Si luego se considera el mensaje m' = $m \,||\, (m \oplus t)$, se observa que el par <m',t> verifica siempre, quebrando así la seguridad del esquema.

Para mensajes de longitud variable, la construcción básica de CBC-MAC debe entonces modificarse. Algunas posibles opciones que se prueba que son seguras son:

1. Aplicar la función seudoaleatoria (block cipher) a la longitud del mensaje de entrada como para obtener una clave $k_l := F_k(l)$. Luego, calcular el CBC-MAC básico usando la

clave k_l. Esto asegura que claves diferentes se usan para autenticar mensajes de diferentes longitudes.

2. Agregar como prefijo al mensaje su longitud $|m|$, codificada como un string de n bits y luego calcular el CBC-MAC básico en el mensaje resultante. (Agregarlo al final **no es seguro**)
3. Cambiar el esquema como para que la generación de claves elija dos claves diferentes $k_1 \leftarrow \{0,1\}^n$ y $k_2 \leftarrow \{0,1\}^n$. Luego, para autenticar el mensaje m, primero calcular el CBC-MAC básico de m usando k_1 y se obtiene de resultado t. Emitir $\hat{t} := F_{k_2}(t)$. En este caso, la ventaja está en que no es necesario saber la longitud del mensaje por anticipado. Su desventaja es que requiere dos claves. Pero se puede almacenar una sola clave k y luego derivar $k_1 = F_k(1)$ y $k_2 = F_k(2)$ al comienzo del cálculo.

d. ¿Para qué sirve la "Transformación de Merkle-Darmgard"?

La transformación de Merkle-Darmgard sirve para construir funciones de hash resistentes a colisiones. Esta metodología habilita una conversión de cualquier función de hash de longitud fija a una función de hash de longitud variable mientras mantiene la propiedad de ser resistente a colisiones.
Se define con la siguiente construcción:

Sea $\pi = $ **(Gen, h)** una **función de Hash resistente a colisiones de longitud fija para entradas de longitud** $2l(n)$ y con longitud de salida $l(n)$. Se construye una función de hash de longitud variable **(Gen, H)** de la siguiente manera:
Gen : se mantiene igual.
H: dada como entrada una clave s y un string $x \in \{0,1\}^*$ de longitud $L < 2^{l(n)}$, hacer lo siguiente:

1) Establecer $B := \left\lceil \frac{L}{l} \right\rceil$ (número de bloques en x. Completar x con ceros como para que la longitud sea múltiplo de $l(n)$. Dividir (Parse) el resultado como una secuencia de bloques de l bits $x_1,\ldots,x_B$. Establecer $x_{B+1} := L$, donde L se codifica usando exactamente $l(n)$ bits.
2) Establecer $z_0 := 0^{l(n)}$
3) Para $i = 1,\ldots,B+1$ calcular $z_i := h^s(z_{i-1} \| x_i)$
4) Emitir como salida z_{B+1}

La seguridad de la transformación de Merkle – Damgard está dada por la seguridad del h^s subyacente. Es decir, una colisión en H^s sólo puede ocurrir si hay una colisión en la h^s subyacente.

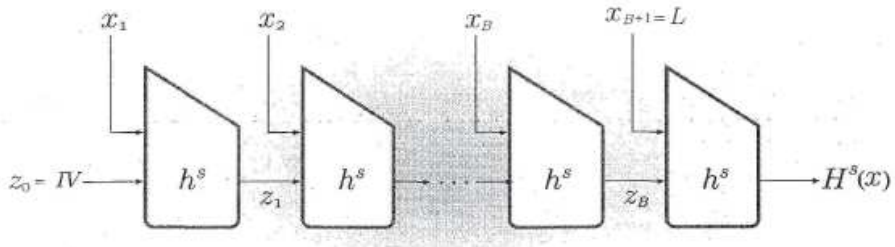

FIGURE 4.2: The Merkle-Damgård transform.