Capítulo 4 – Message Authentication Codes and Collision – Resistant Hash Functions.

1. Comunicación segura e integridad de mensajes.

En muchos casos es de igual o mayor importancia garantizar **integridad de mensajes** (o autenticación de mensajes) en el sentido que cada participante debería poder identificar cuándo un mensaje recibido es exactamente el mensaje enviado por el otro participante.
Ejemplo: Un supermercado envía un mail por orden de compra de x productos.

2. Cifrado vs Autenticación de mensajes.

Los objetivos de privacidad y de integridad son diferentes. Las técnicas y herramientas para conseguirlos también lo son.
Encryption(cifrado) no provee (en general) integridad. Y no debería ser usado para lograr autenticación de mensajes.
Encryption(cifrado) debería usarse en conjunto con otras técnicas para alcanzar autenticación de mensajes.

Cifrado usando stream ciphers

Dado un mensaje cifrado con stream cipher $c := Enc_k(m) = G(k) \oplus m$, donde G es un generador seudoaleatorio, es posible modificar el cifrado c', tal que $m' = Dec_k(c')$ es el mismo que m pero con algunos bits invertidos.

Cifrado usando block ciphers

El mismo ataque anterior podría darse con los esquemas de cifrado en modo OFB o Counter, que cifran mensajes mediante una operación XOR con un stream seudoaleatorio.
Es decir que, aún usando cifrado CPA-Secure, no se garantiza que el mensaje no sea alterado.
En modo ECB, invertir un bit en el i-ésimo bloque de cifrado afecta sólo el pésimo bloque de texto plano. Cambiar ese único bloque podría representar un ataque dañino. Además, el orden de los bloques podría ser cambiado.
En modo CBC, invertir el j-ésimo bit del IV, cambia sólo el j-ésimo bit del primer bloque de mensaje , todos los bloques restantes permanecen inalterados.
Todos los esquemas de cifrado vistos tienen la propiedad que cualquier posible cifrado se corresponde con algún mensaje válido.

3. Message Authentication Codes – Definición.

Son mecanismos destinados a evitar que un adversario modifique un mensaje sin que los participantes en la comunicación lo detecten.

Sintaxis de un Message Authentication Code.

Definición. Un **Message Authentication Code (MAC)** es una tupla de **algoritmos probabilísticos que corren en tiempo polinómico (PPT)** **(Gen, Mac, Vrfy)** tales que:

1) **Gen** = input: el parámetro de seguridad 1^n
   output: una **clave k**. Elige en forma uniforme y aleatoria. Se asume $|k| \geq n$
   $k \leftarrow Gen(1^n)$
2) **Mac** = (**tag generation algorithm**)
   input: **clave k** y **texto plano** $m \in \{0,1\}^*$
   output: un **tag t**
   $t \leftarrow Mac_k(m)$
3) **Vrfy** = (**verification algorithm**)
   input: **clave k, tag t y mensaje m.**
   output: un **bit b**.
   $b := Vrfy_k(m,t)$ **(es determinístico)**
   Si b = 1, se dice que el mensaje es **válido**.
   Si b = 0, se dice que el mensaje es **inválido**.

Se requiere que para todo n, k, m $Vrfy_k(m, Mac_k(m)) = 1$

Si **(Gen, Mac, Vrfy)** es tal que para todo k, $Mac_k(m)$ está sólo definido para mensajes $m \in \{0,1\}^{l(n)}$, entonces se dice que **(Gen, Mac, Vrfy)** es un **MAC de longitud fija**, para mensajes de longitud l(n).

El emisor envía **<t,m>**

Seguridad de los Message Authenticacion Codes.

Se define el experimento $\text{MAC} - \text{forge}_{A,\pi}(n)$: para **Message Authentication Code** $\pi = (Gen, Mac, Vrfy)$, **adversario A** y cualquier **valor n** como parámetro de seguridad:

1) Se genera una **clave k** mediante **Gen(1^n).**
2) Se entrega 1^n al **Adversario A** y acceso a **Mac_k.**
   El adversario emite un par **<m,t>.** Sea **Q** el conjunto de todas las preguntas que A hace al oráculo **Mac**.
3) La salida del experimento es 1 (ÉXITO) si y sólo si:
   (1) **Vrfyk(m,t) = 1** y
   (2) **$m \notin Q$**

Definición Message Authentication Code $\pi = (Gen, Mac, Vrfy)$ **es existentially unforgeable under an adaptive chosen-message attack,** si para todo **adversario PPT A** existe una función despreciable **negl** tal que:

$$\Pr[\text{MAC} - \text{forge}_{A,\pi}(n) = 1] \leq negl(n)$$

¿Es la definición demasiado exigente?

Lo es:

1) El adversario puede requerir etiquetas MAC para cualquier mensaje de su elección.
2) Se considera que el adversario ha quebrado el esquema si puede emitir un tag válido sobre cualquier mensaje no autenticado previamente.

Ataques de Replay

Un MAC no puede proteger contra ataques de repetición de mensajes, ya que la definición de un MAC no incorpora ninguna noción de estado en el algoritmo de verificación.
Se deja a aplicaciones de capas superiores la protección contra estos ataques.
Técnicas posibles:

1) ***Números de secuencia***:
   A cada mensaje se le asigna un número de secuencia (i) y el tag se calcula sobre la concatenación $(i \| m)$.
   Desventaja: el receptor debe almacenar todos los números de secuencia que se han recibido previamente.
2) ***Time-stamps***:
   El emisor adjunta el tiempo actual al mensaje. Cuando el receptor obtiene el mensaje, chequea si el tiempo actual del mensaje está dentro de un margen aceptable de tiempo.
   Desventaja: Requiere relojes sincronizados en emisor y receptor.

4. Construyendo Message Authentication Codes Seguros.

Se usan Funciones Seudoaleatorias.

Sea F una **función seudoaleatoria**. Se define un MAC $\pi = (Gen, Mac, Vrfy)$, para **mensajes de longitud n**, tal que:

1) **Gen** = Elige en forma uniforme y aleatoria **k**$\leftarrow \{0,1\}^n$
2) **Mac** = dada una **clave k** y **un mensaje** $m \in \{0,1\}^n$, se emite la etiqueta **t:=** $F_k(m)$ (Si $|m| \neq |k|$, no emite nada)
3) **Vrfy** = input: **clave k**, **un mensaje** $m \in \{0,1\}^n$, y una **etiqueta t**$\in \{0,1\}^n$,
   output: 1 si y sólo si **t=** $F_k(m)$ (Si $|m| \neq |k|$, emite 0)

es un **Message Authentication Code** $\pi = (Gen, Mac, Vrfy)$ **de longitud fija, para mensajes de longitud fija n, que es existentially unforgeable under an adaptive chosen-message attack** siempre que **F** sea una función seudoaleatoria.

Extensión a Mensajes de Longitud Variable.

A partir de un MAC $\pi' = (Gen', Mac', Vrfy')$, para **mensajes de longitud n**
Sugerencias:

1) ***XOR entre los bloques y luego autenticar el resultado***:
   $t := Mac'_k(\oplus_i m_i)$
   Problema: 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.
2) ***Autenticar cada bloque por separado***:

$t_i := Mac'_k(m_i)$ y emitir $\langle t_1, t_2, ..., t_d \rangle$

Esto evita que un adversario envpia bloques no autenticados previamente sin ser detectados.
Pero:
Problema: El adversario puede cambiar el orden de los bloques y calcular un tag válido sobre ellos (ej. $m_d, ..., m_2, m_1$

3) *Autenticar cada bloque junto con un número de secuencia*:

Calcular $t_i := Mac'_k(i \| m_i)$ y emitir $\langle t_1, t_2, ..., t_d \rangle$

Problema: El adversario puede eliminar bloques al final del mensaje, o mezclar bloques de diferentes mensajes (si obtiene $\langle t_1, t_2, ..., t_d \rangle$ y $\langle t'_1, t'_2, ..., t'_d \rangle$ para los mensajes $m = m_1, ..., m_d$ y $m' = m'_1, ..., m'_d$ puede emitir un tag válido $\langle t_1, t'_2, t_3, t'_4 ... \rangle$ para el mensaje $m_1, m'_2, m_3, m'_4 ...$

Sea un MAC π'= **(Gen', Mac', Vrfy')** , para **mensajes de longitud n**, Se define un MAC π = **(Gen, Mac, Vrfy)** , tal que:

1) **Gen** = Es idéntica a **Gen'**

2) **Mac** = dada una **clave k** $\in \{0,1\}^n$ y **un mensaje** $m \in \{0,1\}^*$, **de longitud** $l < 2^{\frac{n}{4}}$, partir m en d bloques, cada uno de longitud $\frac{n}{4}$ (el último bloque se completa con 0 si es necesario). Luego, elegir un identificador aleatorio $r \leftarrow \{0,1\}^{n/4}$

Para $i = 1, ..., d$, calcular $t_i \leftarrow Mac'_k(r \| l \| i \| m_i)$, donde i y l son unívocamente codificados como strings de longitud $\frac{n}{4}$. Finalmente, emitir el tag $\langle r, t_1, t_2, ..., t_d \rangle$

Vrfy = dada una **clave k** , **un mensaje** $m \in \{0,1\}^*$, **de longitud** $l < 2^{\frac{n}{4}}$ y una **etiqueta t** = $\langle r, t_1, t_2, ..., t_{d'} \rangle$, partir m en d bloques, cada uno de longitud $\frac{n}{4}$ (el último bloque se completa con 0 si es necesario)

output: 1 si y sólo si **d' = d** y $Vrfy'_k(r \| l \| i \| m_i, t_i) = 1, \forall i, 1 \le i \le d$

Si π' es un MAC segura de longitud fija para mensajes de longitud n, entonces esta construcción es un MAC que es existentially unforgeable under an adaptive chose-message attack.

Repeat: con este nombre denotamos el evento de que un mismo identificador de mensaje aparece en dos de las etiquetas que retorna el oráculo MAC en el experimento $\text{MAC} - \text{forge}_{A,\pi}(n)$:.

Forge: con este nombre denotamos el evento de que al menos uno de los bloques $r \| l \| i \| m_i$ nunca fue previamente autenticado por el oráculo, aunque $Vrfy'_k(r \| l \| i \| m_i, t_i) = 1$.

Hay una función despreciable negl con **Pr[Repeat]** $\le negl(n)$

$$\Pr[\text{MAC} - \text{forge}_{A,\pi}(n) = 1 \wedge \overline{\text{Repeat}} \wedge \overline{\text{Forge}}] = 0$$

5. *CBC - MAC*

La construcción anterior, es muy ineficiente: para calcular una etiqueta en un mensaje de longitud $l \cdot n$, es necesario aplicar el block cipher $4l$ veces, la etiqueta MAC es de una longitud de $(4l+1) \cdot n$ bits.
La construcción CBC-MAC es similar al modo de cifrado CBC y se usa mucho.

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**$\leftarrow \{0,1\}^n$

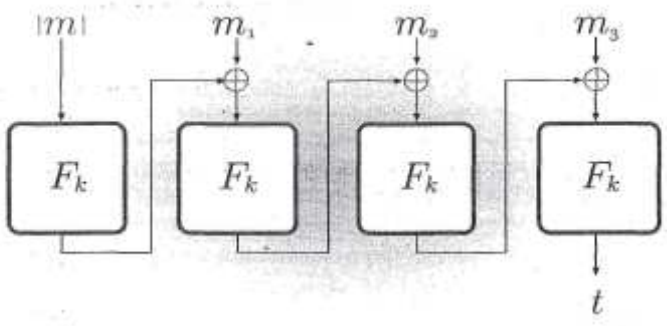
2) **Mac** = dada una **clave k** y **un mensaje** m **,** de longitud $l(n)\cdot n$, hacer lo siguiente:

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 $t \stackrel{?}{=} Mac_k(m)$



CBC - MAC vs CBC - mode encryption

Hay dos diferencias:

1. CBC-mode encryption usa un IV aleatorio y esto es crucial para obtener seguridad.
   CBC-MAC no usa IV, sino un valor fijo $t_0 := 0^n$ y esto es crucial para obtener seguridad.
2. En CBC-mode encryption todos los bloques c_i son emitidos por el algoritmo de cifrado como parte del cifrado, mientras que en CBC-MAC sólo el último bloque es lo que se emite. Si CBC-MAC emitiera todos los bloques ya no sería seguro.

Es crucial comprender la construcción de CBC-MAC. Hay librerías que proveen a los programadores con una "función CBC" pero no distingue entre el uso de esta función para MAC y para cifrado.

Secure CBC - MAC para mensajes de longitud variable

Para obtener una versión de CBC-MAC para mensajes de longitud variable hay tres opciones seguras:

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 $\hat{t} := F_{k_2}(t)$

La tercer opción tiene como ventaja 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.

6. Funciones de Hash Resistentes a Colisiones

Funciones de Hash Resistentes a Colisiones: Son funciones que comprimen strings de longitud arbitraria en strings de longitud fija.

Es obligatorio **evitar colisiones** en su uso en el ámbito de la criptografía.

6.1. Definiendo Resistencia a Colisiones

Una **colisión** en una función H es un par de inputs diferentes x y x', tales que $H(x) = H(x')$. En este caso decimos que x y x' colisionan bajo H.

Una función H es **resistente a colisiones** si no es factible (infeasible) para un algoritmo PPT encontrar una colisión en H.

Como las funciones H tiene un dominio infinito pero un rango finito, las **colisiones** van a existir.

La función H, es una función que toma como entrada una clave s y un string x, y emite un string de salida $H^s(x) = H(s,x)$. El requisito es que debe ser difícil (hard) encontrar una colisión en H^s para un s elegido aleatoriamente.
La clave s no es una clave criptográfica habitual, y hay dos diferencias con las claves criptográficas:

1. No todos los strings necesariamente se corresponden con claves válidas, y por lo tanto la clave s debería ser generada por un algoritmo Gen en lugar de ser elegida uniforme y aleatoriamente.
2. Esta clave s no se guarda en secreto. Por eso lo escribimos como H^s en lugar de H_s

Definición. Una **Función de Hash** es una par de **algoritmos probabilísticos que corren en tiempo polinómico (PPT) (Gen, H)** tales que:

- **Gen**: es un algoritmo probabilística que toma como entrada un parámetro de seguridad 1^n y emite una clave s. Se asume que 1^n es implícito en s
- Existe un polinomio l tal que H toma como entrada s y un string $x \in \{0,1\}^*$ y emite un string $H^s(x) \in \{0,1\}^{l(n)}$ (donde n es el valor del parámetro de seguridad implícito en s)

Si H^s está definido solamente para entradas $x \in \{0,1\}^{l'(n)}$ y $l'(n) > l(n)$, entonces diremos que **(Gen, H) es una función de Hash de longitud fija para entradas de longitud** $l'(n)$

Se define el experimento para encontrar colisiones $Hash - coll_{A,\pi}(n)$: para **hash function** $\pi =$ **(Gen, H), adversario A** y cualquier **valor n** como parámetro de seguridad:

1) Se genera una **clave s** mediante **Gen(1^n).**
2) Se entrega **s** al adversario y éste emite x, x'.
3) La salida del experimento es 1 (ÉXITO) si y sólo si: $x \neq x'$ y $H^s(x) = H^s(x')$. En ese caso diremos que el adversario encontró una **colisión**.

Definición Una función de hash $\pi =$ (Gen, H) es resistente a colisiones si para todo adversario PPT existe una función despreciable negl tal que :

$$\Pr[Hash - coll_{A,\pi}(n) = 1] \leq negl(n)$$

Se tratará como conceptos equivalentes a: $H, H^s, \Pi = (Gen, H)$

6.2. Nociones de Seguridad más débiles.

La resistencia a colisiones es un requisito muy fuerte y muy difícil de alcanzar. Sin embargo, en algunas aplicaciones alcanza con requisitos más relajados.
Se pueden considerar tres niveles de seguridad:

1. **Resistencia a colisiones**. El nivel más exigente.
2. **Resistencia a segundas preimágenes**. Si dados s y x no es factible, para un adversario PPT A, encontrar $x' \neq x$ tal que $H^s(x) = H^s(x')$
3. **Resistencia a preimagen**. Si dados s y $y = H^s(x)$, para un valor x elegido aleatoriamente, no es factible para un adversario PPT encontrar un valor x' tal que $H^s(x') = y$. Esto también significa que H^s es una función de una sola vía.

Toda función de hash que es resistente a colisiones es resistente a segundas preimágenes.
Toda función de hash que es resistente a segundas preimágenes es resistente a preimagen.

Estos tres requisitos de seguridad forman entonces una jerarquía, con cada definición implicando la siguiente.

6.3. Un ataque de "Cumpleaños" genérico.

Ataques de cumpleaños en funciones de Hash - resumen.
Este ataque implica que hay una longitud mínima de la salida necesaria para una función de hash para que sea potencialmente segura contra adversarios ejecutando en un cierto tiempo.

El ataque funciona de la siguiente manera:

Se eligen q inputs arbitrarios $x_1, x_2, ..., x_q \in \{0,1\}^{2l}$ y se calculan $y_i := H(x_i) \forall i$, y se determina si algún par de valores y_i son iguales.

¿cuál es la probabilidad de que este algoritmo encuentre una colisión?

Si $q > 2^l$, esto ocurre con probabilidad 1.

Este problema se conoce como problema del cumpleaños: si q personas están en una habitación ¿cuál es la probabilidad de que dos de ellas tengan la misma fecha de cumpleaños? Si y_i representa el día del cumpleaños de la persona i, entonces tenemos $y_1, y_2, ..., y_q \leftarrow \{0,...,365\}$. Buscamos entonces para que valores $i \neq j$ resulta que $y_i = y_j$. Si hay sólo 23 personas en la habitación, la probabilidad de que dos de ellas cumplan los años el mismo día es de más de 0,5.

Cuando $q = \Theta\left(2^{\frac{l}{2}}\right)$, la probabilidad de que $y_i = y_j$ es de 0,5.

Si la longitud de salida es de l bits, entonces el ataque del cumpleaños encuentra una colisión con gran probabilidad usando evaluaciones de funciones de hash $O(q) = O\left(2^{\frac{l}{2}}\right)$, una colisión puede ser encontrada en tiempo $O\left(l \cdot 2^{\frac{l}{2}}\right)$

Esto determina que una función de hash resistente a colisiones, en la práctica, necesita tener una salida que sea mayor de 128 bits. De todas maneras, esta es una condición necesaria, pero no suficiente.

Ataques de cumpleaños mejorados.

6.4. La transformación de Merkle - Damgard

Metodología 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 "full-fledged" 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 $x_1, ..., x_B$. Establecer $x_{B+1} := L$, donde L se codifica usando exactamente $l(n)$ bits.
2) Establecer $z_0 := 0^{l(n)}$
3) Para $i = 1, ..., B+1$ calcular $z_i := h^s(z_{i-1} \| x_i)$
4) Emitir como salida z_{B+1}

El vector de inicialización.

El valor z_0 puede ser reemplazado por cualquier constante y se conoce como Vector de Inicialización.

La seguridad de la transformación de Merkle - Damgard.

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

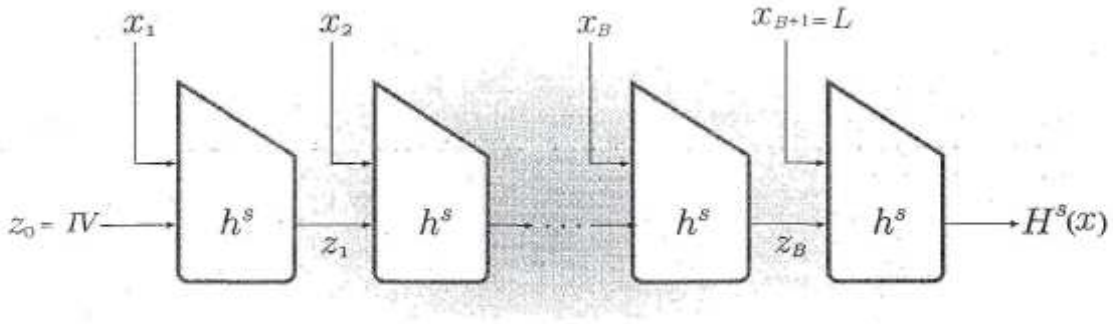


FIGURE 4.2: The Merkle-Damgård transform.

6.5. Funciones de Hash Resistentes a Colisiones en la práctica.

Las construcciones de funciones de hash resistentes a colisiones vienen en dos formas:
- provably-secure constructions based on certain number-theoretic assumptions
- highly-efficient constructions that are more euristic in nature.

Estudiamos las últimas.
Las funciones de hash en la práctica generalmente no usan clave. Por lo tanto no hay algoritmo Gen.
Las funciones de hash libres de colisiones deberían tener una longitud de salida de al menos 160 bits, como para que un ataque de cumpleaños lleve un tiempo de 2^{80} hash computations (ver)

Hay dos funciones de hash populares: MD5 y SHA-1
MD5 no es más segura y NO debería usarse en ninguna aplicación que requiera resistencia a colisiones.
Tanto MD5 como SHA-1 primero definen una función de compresión que comprime la entrada de longitud fija en una longitud relativamente pequeña. Luego se aplica la transformación de Merkle-Damgard a la función de compresión para obtener una función de hash resistente a colisiones para entradas de longitud arbitraria. La salida de MD5 es de 128 bits y la de SHA-1 es de 160 bits. Para MD5 un ataque de cumpleaños demanda 2^{64} hash computations, mientras que para SHA-1 demanda 2^{80} hash computations.
En 2004 un grupo de criptoanalistas chinos presentaron un ataque al MD5. También al SHA-1, pero muestran que el ataque a éste demanda 2^{69} hash computations. Por eso se prefiere apuntar a funciones de hash de 256 o 512 bits.

7. NMAC y HMAC

Están basadas en funciones de hash resistentes a colisiones construidas usando la transformación de Merkle –Damgard a alguna función de compresión subyacente.

Se denota con $H^s_{IV}(x)$ al cálculo de la Construcción con entrada x, clave s, y con z_0 establecido en el valor $IV \in \{0,1\}^l$

7.1. Nested MAC (NMAC)

Sea H una función de hash construida usando Merkle-Damgard aplicada a la función de compresión h. Diremos que n denota la longitud de H y h, y seguiremos asuiendo que h comprime su entrada a la mitad.
El primer paso en la construcción de NMAC es definir versiones secretly-keyed de las funciones de compresión y de hash.

Sea s una clave fija, no secreta y definamos una versión secretly-keyed de h^s como $h^s_k(x) \overset{def}{=} h^s(k \parallel x)$

Se define H^s_k como una función de hash obtenida estableciendo $IV = k$ en la construcción de Merkle.

Entonces obtenemos $z_1 = h^s(k \parallel x_1) = h^s_k(x_1)$

Dada $\left(\tilde{Gen}, h\right)$ una función de resistencia de colisiones de longitud fija, sea $\left(\tilde{Gen}, H\right)$ el resultado de aplicarle Markle
NMAC define un MAC de la siguiente manera:

- **Gen**: para una entrada 1^n, ejecutar $\tilde{Gen}(1^n)$ para obtener la clave s. También elegir $k_1, k_2 \leftarrow \{0,1\}^n$ al azar. Se emite la clave (s, k_1, k_2)

- **Mac**: para una entrada (s, k_1, k_2), un mensaje $m \in \{0,1\}^*$, emitir la etiqueta $t := h^s_{k1}(H^s_{k2}(m))$
- **rfy**: para una entrada (s, k_1, k_2), un mensaje $m \in \{0,1\}^*$ y una etiqueta t, emite 1 si y sólo si $t \stackrel{?}{=} Mac_{s,k1,k2}(m)$

La seguridad de NMAC reside en la suposición de que h^s_k con clave k constituye una MAC segura.

Security Assumptions

El suponer que $\left(\tilde{Gen}, h\right)$ da una MAC segura NO es consecuencia de suponer que es resistente a colisiones.

La clave s no es necesario que permanezca en secreto.

La clave k_2 no es necesaria una vez que se asume que $\left(\tilde{Gen}, h\right)$ es resistente a colisiones.

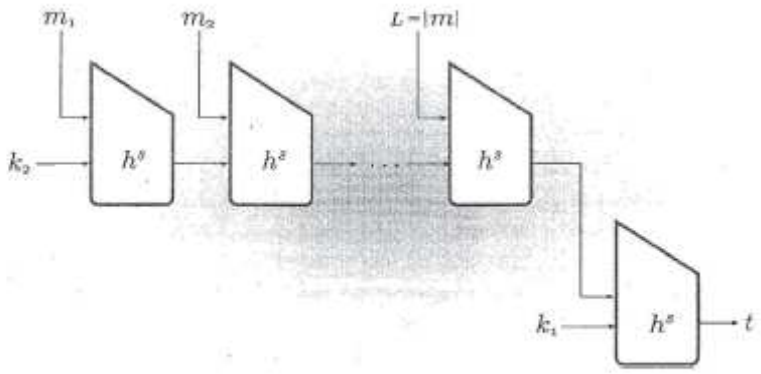


FIGURE 4.3: NMAC.

7.2. HMAC

Una desventaja de NMAC es que el IV de la función H de hash debería ser modificado. En la práctica esto puede causar complicaciones.
HMAC resuelve esto usando claves distintas para la compresión y para la función de hash. Otra diferencia es que HMAC usa una sola clave secreta en lugar de dos claves secretas.

Sea H y h como para NMAC.
Diremos que n denota la longitud de H y h, y seguiremos asuiendo que h comprime su entrada a la mitad.
Sea IV un valor fijo para el vector de inicialización, que se asume que está fuera del control de las partes honestas. Asumimos que cuando $x \in \{0,1\}^n$ entonces el cálculo de $H^s_{IV}(x)$ involucra sólo una invocación a la función de compresión h^s.

Dada $\left(\tilde{Gen}, h\right)$ una función de resistencia de colisiones de longitud fija, sea $\left(\tilde{Gen}, H\right)$ el resultado de aplicarle Markle. Sea IV, opad y ipad constantes fijas de longitud n.
HMAC define un MAC de la siguiente manera:

- **Gen**: para una entrada 1^n, ejecutar $\tilde{Gen}(1^n)$ para obtener la clave s. También elegir $k \leftarrow \{0,1\}^n$ al azar. Se emite la clave (s, k)
- **Mac**: para una entrada (s, k), un mensaje $m \in \{0,1\}^*$, de longitud L, emitir la etiqueta $t := H^s_{IV}\left((k \oplus opad) \| H^s_{IV}((k \oplus ipad) \| m)\right)$

- rfy: para una entrada (s,k), un mensaje $m \in \{0,1\}^*$ y una etiqueta t, emite 1 si y sólo si $t \stackrel{?}{=} Mac_{s,k}(m)$

HMAC usa dos constantes: opad y ipad. Son dos strings de longitud n y se definen:
- opad consiste de un byte "0x36" repetido tantas veces como sea necesario.
- Ipad consiste de un byte "0x5C" repetido tantas veces como sea necesario.

Es posible ver a HMAC como una variante de NMAC:
$H^s_{IV}((k \oplus opad) \| H^s_{IV}((k \oplus ipad) \| m)) = H^s_{k1}(H^s_{k2}(m)) = h^s_{k1}(H^s_{k2}(m))$ por ser de longitud de un bloque.

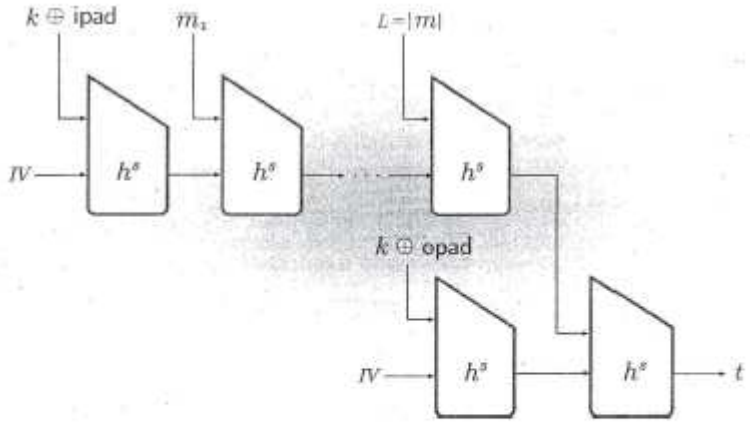


FIGURE 4.4: HMAC.

fine

$$G(k) \stackrel{def}{=} h^s(IV \| (k \oplus \text{opad})) \| h^s(IV \| (k \oplus \text{ipad})) = k_1 \| k_2. \qquad (4.4)$$

HMAC en la práctica
Es un estandar industrial.
Es altamente eficiente y fácil de implementar.

8. *Construyendo CCA – Secure Encryption Schemes*

Emisor y receptor comparten dos claves: una para el esquema de cifrado CPA-secure y la otra para el MAC.

Dado $\pi_E = (Gen_E, Enc, Dec)$, un esque a de cifrado de c ave privada y sea $\pi_M = (Gen_M, Mac, Vrfy)$ un Message Authentication Code MAC), se define un esquema de cifrado (Gen', Enc', Dec') de la siguiente manera:

1) Gen, = dada la entrada 1^n, se ejecutan $k_1 \leftarrow Gen_E(1^n)$ y $k_2 \leftarrow Gen_M(1^n)$
) Enc, = dada como entrada una clave (k_1, k_2) y un texto plano m, calcular c←Enc$_{k1}$) y t←Mac$_{k2}$ c) y emitir como salida $\langle c, t \rangle$
3) Dec, = dada como entrada una clave (k_1, k_2) y un cifrado $\langle c, t \rangle$, primero chequear si rfy$_{k2}$ c t)=1 Si verifica, entonces emitir como salida Dec$_{k1}$ c) Sino, emitir $\perp$ (fallo)

$\pi_M = (Gen_M, Mac, Vrfy)$ tiene etiquetas nicas si para todo k y para todo m, hay un único valor t para el cual rfy$_k$ t)=1

Si π_E es un esque a de cifrado de c ave privada CPA-secure, y π_M un Message Authentication Code MAC) seguro con etiquetas únicas, ,entonces la construcción anterior es CCA-secure.

The role of unique tags.

Si el esquema π_M no tiene etiquetas únicas, entonces la construcción anterior puede no ser CCA-secure. Podria ser fácil modificar una etiqueta válida t sobre un valor c, en una etiqueta diferente t' sobre el

mismo valor. Si el cifrado del desafío es <c, t> entonces un adversario puede enviar <c,t'> al oráculo y así probablemente conocer el texto plano m_b.
Una restricción más débil: alcanza con que un adversario no pueda encontrar una etiqueta válida t' en un mensaje m previamente autenticado.

CCA - security and real - life implications

La construcción alcanza tanto privacidad como autenticación de mensajes.

9. Obteniendo Privacidad y Autenticación de Mensajes.

Hay tres enfoques comunes para combinar cifrado y autenticación de mensajes.

Sea k_1 una clave de cifrado y sea k_2 una clave MAC.

Los tres enfoques son:

1. ***Cifrar y autenticar.***
   En este método, el cifrado y la autenticación se calculan por separado.
   El emisor transmite $\langle c,t \rangle$, donde: $c \leftarrow Enc_{k1}(m)$ y $t \leftarrow Mac_{k2}(m)$
   ES INSEGURO: existen esquemas de cifrado y MAC seguros para los que esta combinación es insegura.
2. ***Autenticar y luego cifrar.***
   Primero se calcula la etiqueta t, y luego el mensaje y la etiqueta se cifran juntos.
   El emisor transmite c ,donde $t \leftarrow Mac_{k2}(m)$, y $c \leftarrow Enc_{k1}(m \| t)$.
3. ***Cifrar y luego autenticar.***
   Primero se cifra el mensaje m y luego se calcula una etiqueta MAC sobre el mensaje cifrado. Esto es, el mensaje es un par $\langle c,t \rangle$, donde $c \leftarrow Enc_{k1}(m)$, y $t \leftarrow Mac_{k2}(c)$

Sólo Privacidad vs Privacidad e Integridad de Mensajes

Es buena práctica cifrar y autenticar siempre.
La falta de integridad puede llevar muchas veces a una brecha en la privacidad, como se vio en los casos de chosen-ciphertext attacks.

Requerimientos de Seguridad

Definición: Sea $\pi_E = (Gen_E, Enc, Dec)$, un esque a de cifrado de c ave privada y sea $\pi_M = (Gen_M, Mac, Vrfy)$ un Message Authentication Code MAC).

Un esque a de trans ision de ensa es $\pi' = (Gen', EncMac', Dec')$ derivado de la combinación de π_E y π_M es una tupla de algoritmos que operan de la siguiente manera:

- Gen›: toma una entrada 1^n, y ejecuta $Gen_E(1^n)$ y $Gen_M(1^n)$ para obtener k_1 y k_2. La clave es (k_1, k_2)
- EncMac›: el algoritmo de transmisión de mensajes, toma como entradas (k_1, k_2), un mensaje m y emite un valor c que se deriva de aplicar alguna combinación de $Enc_{k1}(.)$ y $Mac_{k2}(.)$
- Dec›: toma como entrada (k_1, k_2) y un valor transmitido c, y aplica alguna combinación de $Dec_{k1}(.)$ y $Vrfy_{k2}(.)$. La salida es o bien el texto plano m, o bien el símbolo $\perp$ que denota error.

El esquema será correcto si $Dec'_{k1,k2}(EncMac'_{k1,k2}(m)) = m$

Experimento $Auth_{A,\pi'}(n)$, sobre esque a de trans ision π' y adversario A.

1. Se genera una clave aleatoria (k_1, k_2) ejecutando Gen'.
2. El adversario A recibe 1^n y acceso al oráculo de EncMac›). El adversario eventualmente emite c. Sea Q el conjunto de todas las consultas del adversario A al oráculo.
3. Sea $m := Dec'_k(c)$. La salida de este experimento se define como 1 si y sólo si: $m \neq \perp$ y $m \notin Q$

Un esque a de trans ision π' logra una co unicacion autenticada si para todo adversario PPT A, existe una función negl tal que:

$$\Pr[Auth_{A,\pi}(n) = 1] \leq negl(n)$$

Un esque a de trans ision π' es seguro si es un esquema de cifrado CCA-secure y al mismo tiempo logra una comunicación autenticada.

Cifrar y autenticar

Esta combinación no es necesariamente segura, ya que puede violar la privacidad. Una MAC segura no necesariamente implica privacidad y es posible que la etiqueta del mensaje de a conocer el mensaje completo.

Por ejemplo, si $\pi_M = (Gen_M, Mac, Vrfy)$ es un MAC seguro, también lo es el esquema definido por $Mac'_k(m) = (m, Mac_k(m))$. Por lo tanto, la combinación puede llevar a un esquema que no tiene ni siquiera "indistinguishable encryptions in the presence of an eavesdropper", que es el nivel más básico de seguridad.

Autenticar y luego cifrar

Sea **ransfor**) de la siguiente manera: todo 0 en m es transformado en 00, y todo 1 en m es transformado arbitrariamente en 01 o en 10. El inverso de esta transformación divide en mensaje codificado en pares de bits, y luego asocia 00 a 0, y 10 o 01 a 1. Si se encuentra 11, el resultado es $\perp$

Sea $Enc_k(m) = Enc'_k(Transform(m))$, donde Enc' representa el modo de cifrado counter usando una función seudoaleatoria. Enc es CPA-secure.

La combinación de autenticar y luego cifrar con el esquema de cifrado anterior y cualquier MAC no es seguro contra chosen-ciphertext attack:

> Dado un texto desafío $c = Enc'_{k1}(Transform(m \,|||\, Mac_{k2}(m)))$, el atacante invierte los primeros dos bits del segundo bloque de c (porque el primer bloque de c es un valor inicial del counter) y verifica si el cifrado resultante es válido. (Esto puede hacerlo mediante una consulta al oráculo de desencripción. El adversario sólo necesita saber si el cifrado es válido, sin embargo). Si el primer bit del mensaje m es 1, entonces el cifrado modificado será válido. Esto es porque si el primer bit de m es 1, entonces los pdos primeros bits de Transform(m) son 01 o 10, y al invertirlos se sigue teniendo otra codificación válida de m. Más aún, la etiqueta seguirá siendo válida ya que es´ta aplicada a m y no a la codificación de m. Por otro lado, si el primer bit de m es 0, entoces el cifrado modificado no será válido ya que los dos bits de Transform serían 00 y su cumplemento sería 11.
>
> Este ataque puede ser llevado a cabo sobre cada bit de m en forma separada, resultando en una completa recuperación del mensaje m.

Este contraejemplo demuestra que la combinación autenticar y luego cifrar no siempre es segura. Algunas combinaciones usadas dentro de SSL lo son. Pero no es bueno usar metodologías cuya seguridad depende de implementaciones específicas.

Cifrar y luego autenticar

Sea $\pi_E = (Gen_E, Enc, Dec)$, un esque a de cifrado de c ave privada CPA-secure y sea $\pi_M = (Gen_M, Mac, Vrfy)$ un Message Authentication Code MAC) con etiquetas nicas. Entonces, la combinación $(Gen', EncMac', Dec')$ derivada de aplicar el enfoque de "cifrar y luego autenticar", a π_E, π_M es un esque a de trans ision de ensa es seguro

Transmisiones de mensajes seguras vs CCA - Security

Aunque se usen las mismas construcciones para lograr CCA-security y Secure message transmisión, los objetivos de seguridad en cada caso son diferentes.

En la conformación de la CCA-security, no necesariamente estamos interesados en obtener autenticación de mensaje, sino en asegurar privacidad aún con un adversario fuerte que puede hacer consultas de descifrado.

Al considerar secure message transmission, estamos interesados en obtener CCA-security y integrity.

Es decir, al considerar secure message transmission, se obtiene CCA-security, pero al revés, no siempre.

La necesidad de claves independientes.

Importante: Diferentes objetivos de seguridad deben usar diferentes claves.

Ejemplo:

Sea F una permutación seudoaleatoria fuerte.

Sea $Enc_k(m) = F_k(m \,\|\, r), m \in \{0,1\}^{n/2}, r \leftarrow \{0,1\}^{n/2}$

Sea $Mac_k(c) = F^{-1}{}_k(c)$

Puede demostrarse que este esquema es CPA-secure. Sin embargo, la combinación "cifrar y luego autenticar" aplicada al mensaje m con la misma clave resulta:

$$Enc_k(m), Mac_k(Enc_k(m)) = F_k(m \| r), F^{-1}(F_k(m \| r)) = F_k(m \| r), m \| r$$

y el mensaje m aparece en plano!