Criptografía y Seguridad

Criptografía: MACs y Cifrado Autentificado

Agenda

- Repaso CPA
- Ataques de Texto Cifrado Escogido
- MACs
- Funciones de hash criptograficas
- Cifrado autenticado

Criptosistema simétrico (repaso)

- Es una terna de algoritmos
  - Gen (algoritmo de generación de claves)
  - Enc (cifrado): $Enc_k(m)$
  - Dec (descifrado): $Dec_k(c)$
- Propiedades
  - La salida de Gen define K el espacio de claves.
  - La entrada de Enc define el espacio de mensajes
  - La entrada de Dec define el espacio de mensajes cifrados
  - Para todo m y k válidos: $Dec_k(Enc_k(m))=m$

Ataques de texto plano elegido

- Chosen Plain Text indistinguishability: $CPA_{A,\Pi}$
- Dado un nivel de seguridad n, un adversario A, y un Criptosistema $\Pi(n)$:

1) Se genera una clave $k \leftarrow K$
2) A obtiene $f(x)=Enc_k(x)$ y emite (m_0, m_1)
3) Se genera $b \leftarrow \{ 0, 1 \}$
4) Se calculan $c_i \leftarrow Enc_k(m_b)$ y se le envían a A
5) A emite $b' = \{ 0, 1\}$

- $CPA_{A,\Pi} = 1$ si b = b' (A gana)

Si $Pr[CPA_{A,\Pi}=1] < ½ + neg(n) \Rightarrow \Pi$ es indisting.

Criptosistemas CPA-Secure (Rep.)

- Criptosistemas de flujo con tweaks
  - Utilizan un parámetro extra, llamado IV para evitar reutilizar exactamente la misma semila en el generador
  - El IV es público pero se selecciona aleatoriamente.
- Criptosistemas de bloque
  - Con encadenamiento CBC, Counter, OFB, CFB.
  - Utilizando primitivas de cifrado de bloque seguras

Para reflexionar

- ¿Podríamos utilizarlo para cifrar los sueldos de empleados en una base de datos?
  - Supongamos un criptosistema de flujo

empleado	sueldo
2542	0xEA26969AA3F61CDD9EC68498DF031CA935EFA9C
2678	0xF4933E107178B88D8EE00F40E43A9A3C9D2EDF79
2789	0x155BBBE7A5442CBBCAB8F57DACF7212255F3641C
2890	0x4D9B47D518F2ED7DE04D601F1856767969A328E8

Para reflexionar

- ¿Podríamos utilizarlo para cifrar los sueldos de empleados en una base de datos?

empleado	sueldo
2542	0xEA26969AA3F61CDD9EC68498DF031CA93
2678	0x4D9B47D518F2ED7DE04D601F1856767969A328E8
2789	0x155BBBE7A5442CBBCAB8F57DACF7212255F3641C
2890	0x4D9B47D518F2ED7DE04D601F1856767969A328E8

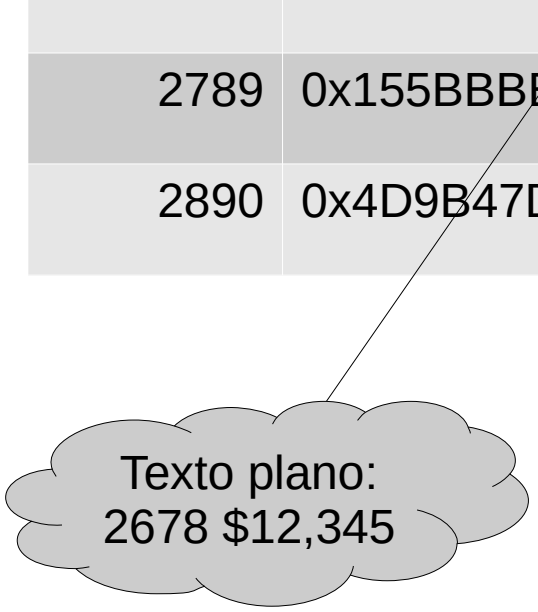
¡Aumento de sueldo!

Supongamos que es un director

Comienza el juego

- Ciframos 'empleado||sueldo'

empleado	sueldo
2542	0xEA26969AA3F61CDD9EC68498DF031CA935EFA9C
2678	0xF4933E107178B88D8EE00F40E43A9A3C9D2EDF79
2789	0x155BBBE7A5442CBBCAB8F57DACF7212255F3641C
2890	0x4D9B47D518F2ED7DE04D601F1856767969A328E8

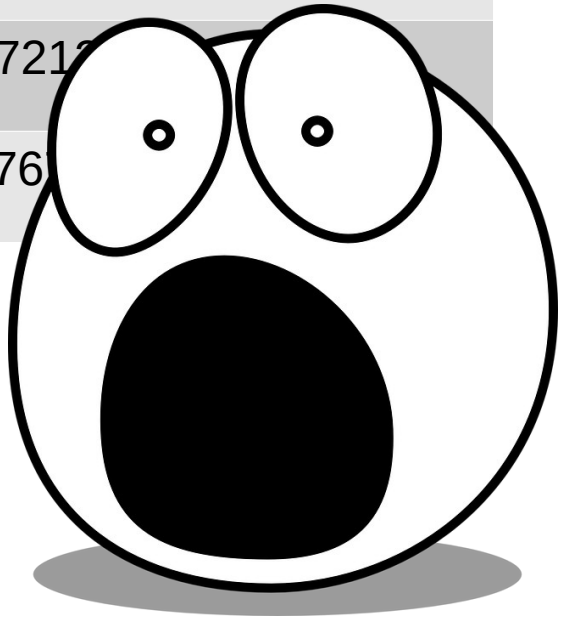


Termina el juego

- Ciframos 'empleado||sueldo'

empleado	sueldo
2542	0xEA26969AA3F61CDD9EC68498DF031CA935EFA9C
2678	0xF4933E107178B88D8EE00F40E43A9A3C9D2F69F0
2789	0x155BBBE7A5442CBBCAB8F57DACF721
2890	0x4D9B47D518F2ED7DE04D601F185676

Texto plano:
2678 $100,000



Backstage

- Se necesita conocer el formato
  - En este caso era IV || empl || sueldo
  - IV = byte[] (12 bytes)
  - empl = int (4 bytes)
  - sueldo = int (4 bytes)

```
0x    F4933E107178B88D8EE00F40    E43A9A3C    9D2EDF79
```

Backstage

- Considerar que:
  - $Enc_k(m) = G(k) \oplus m = c$
  - Definiendo $c' = c \oplus x$
    - $Dec_k(c') = m \oplus x$
- A contar bits:

2EDF79	00101110	11011111	01111001
xor			
12345	00000000	00110000	00111001
xor			
100000	00000001	10000110	10100000
=			
2F69F0	00101111	01101001	11110000

Ataques de texto cifrado escogido

- Chosen Ciphertext Attack: $CCA_{A,\Pi}$
- Dado un nivel de seguridad n, un adversario A, y un Criptosistema $\Pi(n)$:

1) Se genera una clave $k \leftarrow K$
2) A obtiene $f(x)=Enc_k(x)$ y $g(x) = Dec_k(x)$
3) A emite (m_0, m_1)
4) Se genera $b \leftarrow \{0, 1\}$ y se envía $c \leftarrow Enc_k(m_{bi})$
5) A emite b' = { 0, 1} (A no puede ver g(c))

- $CCA_{A,\Pi} = 1$ si b = b' (A gana)

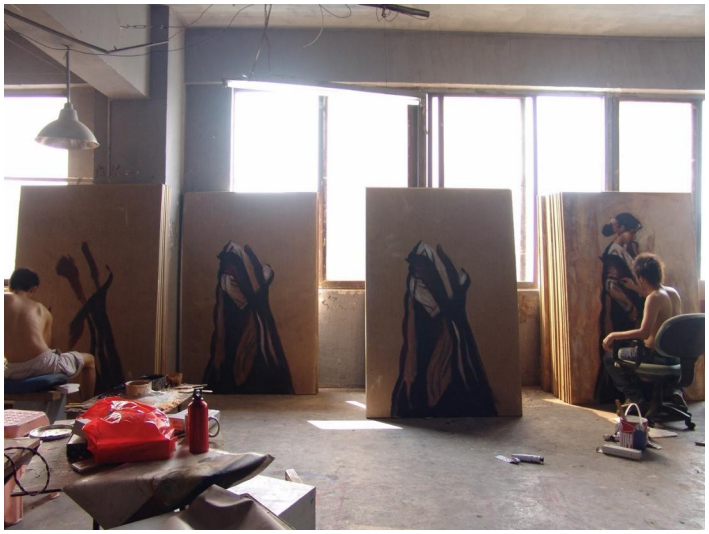
Si $Pr[CCA_{A,\Pi}=1] < ½ + neg(n)$ => Π es indisting.

Ataques de texto cifrado escogido

- Ningún criptosistema visto hasta el momento es CCA-Secure
- Ejercicio: Demostrar que el cifrado de flujo en general no es CCA-Secure
- Ayuda: Considerar m0 = (0...0) y m1 = (1...1) y verificar que consultas se pueden hacer a g(x) al recibir c.

Ataques de texto cifrado escogido

- Es un problema no resuelto por el cifrado solamente.
- Requiere control de integridad
  - Consiste en identificar adulteraciones
  - Primitiva criptográfica: MAC
    - MAC = Message Authentication Code



MAC

- Es una terna de algoritmos
  - Gen (algoritmo de generación de claves)
  - Mac (etiquetado): $t \leftarrow Mac_k(m)$
  - Vrfy (verificación): $b = Vrfy_k(m, t)$
- Propiedades
  - La salida de Gen define K el espacio de claves.
  - Mac es una función no determinística
  - Vrfy retorna 0 (invalido) o 1 (valido)
  - Para todo m y k validos: $Vrfy_k(m, Mac_k(m)) = 1$

Seguridad de un MAC

- Message Authenticaiton Experiment: Mac-forge$_{A,\Pi}$
- Dado un nivel de seguridad n, un adversario A, y un Criptosistema $\Pi(n)$:

1) Se genera una clave k $\leftarrow$ K
2) A obtiene f(x)=Mac$_k$(x)
3) A realiza las evaluaciones que quiera de f(x)

   (Llamese Q al conjunto de las evaluaciones)
4) A emite (m, t)

- Mac-Forge$_{A,\Pi}$ = 1 si Vrfy$_k$(m, t) = 1 y m no pertenece a Q

Seguridad de un MAC

- $\text{Mac-Forge}_{A,\Pi} = 1$ si $\text{Vrfy}_k(m, t) = 1$ y m no pertenece a Q
- Si Pr[Mac-Forge = 1] < neg(n) =>
  - Π es infalsificable ante un ataque de mensaje escogido
- Obervaciones:
  - El adversario puede obtener un MAC para cualquier mensaje que elija
  - Se considera roto el MAC si el adversario puede falsificar cualquier mensaje, independientemente de si tiene sentido o no.

Ejercicio

- Considerar la seguridad de los siguientes Macs:
  - $Mac_k(m) = G(k) \oplus m$ (G(-) gen. Pseudoaleat.)
  - $Mac_k(m) = k \oplus$ first_k_bits(m)
  - $Mac_k(m) = Enc_k(|m|)$ (Enc, CPA-Secure)

Como construir un MAC

- 1ra opción: A partir de una primitiva de cifrado de bloque
- Sea F una función pseudoaleatoria
- CBC-MAC (tamaño fijo de mensajes):
  - Gen: $k \leftarrow \{0, 1\}^n$
  - Mac:
    - sea $m = m_1||m_2||m_3...||m_j$, Sea $t_0 = 00...0$
    - $t_i = F_k(t_{i-1} \oplus m_i)$.
    - $Mac_k(m) = t_j$
  - Vrfy: $Vrfy_k(m, t) = 1 \leftrightarrow t = mac_k(m)$

CBC-MAC

- La Construcción anterior es infalsificable SOLO si se permiten mensajes de una misma longitud

Demostrarlo a través de una prueba de seguridad

CBC-MAC

- La Construcción anterior es infalsificable SOLO si se permiten mensajes de una misma longitud

- Considerar: m_1=A||B, m_2=A

  (A y B tienen el tamaño de un bloque)

- Obtener t_1 y t_2

- Intentar: m_3=A||B||(A xor t_1) t_3=t_2

CBC-MAC

- Extensiones seguras para mensajes arbitrarios:
  - Computar $k' = F_k(|m|)$

    Calcular $T_i = F_{k'}(t_{i-1} \oplus m_i)$. $MAC_k(m) = t_j$
  - Computar $m' = |m| \,||\, m$

    $Mac_k(m) = CBC\text{-}MAC_k(m')$
  - Generar dos claves k_1 y k_2
    - Calcular $t' = CBC\text{-}MAC_{k1}(m)$
    - $t = F_{k2}(t')$

CBC-MAC

- Ejercicio:

Demostrar que este MAC no es seguro

$$Mac_k(m) = CBC\text{-}MAC_k(m')$$

$$m' = m \,||\, |m|$$

(CBC-MAC donde al mensaje se le agrega la longitud como sufijo en lugar de prefijo)

CBC-MAC

- Ejercicio:

$Mac_k(m) = CBC\text{-}MAC_k(m')$

$m' = m \,||\, |m|$

Considerar:

m_1=AAA → m_1'=AAA3

m_2=BBB → m_2'=BBB3

m_3=AAA3CC → m_3'=AAA3CC6

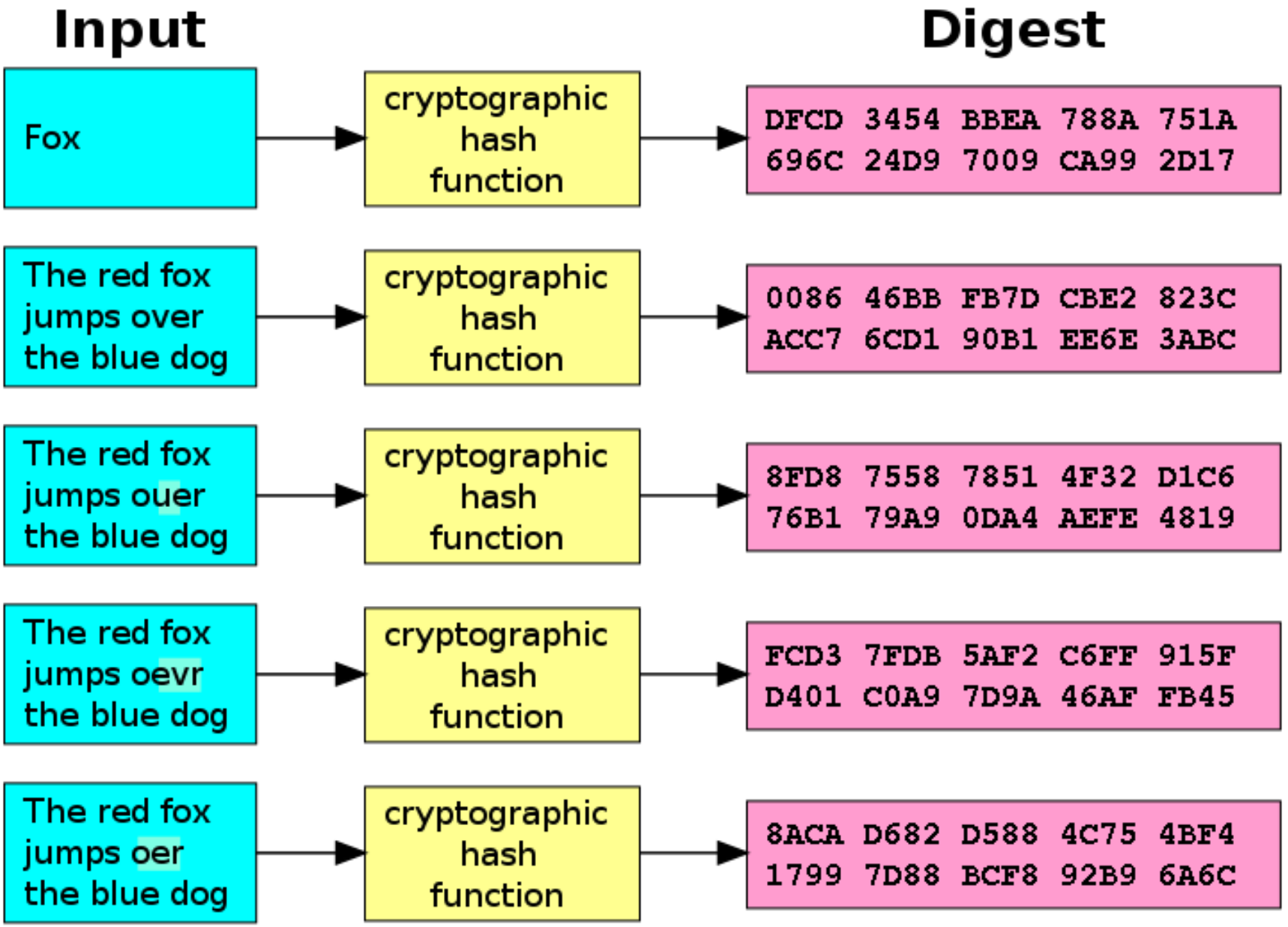
Intentar m=BBB3XC

donde $X=t_1$ xor t_2 xor C

Funciones de hash criptograficas

- Son pares de algoritmos:
  - Gen: $s \leftarrow S$
  - Hash: $h = H^s(m)\ \varepsilon\ \{0,1\}^L$
    - L es la longitud del hash
- Diferencia importante:
  - s no es una clave. Simplemente es un selector.
  - En muchas implementaciones $S = \{ s_0 \}$
    - Solo hay una función de hash en la familia
- Se las llama funciones de resumen
- Analogas a los MACs, pero sin clave

Etiquetadores universales



Colisiones

- Colision:
  - $x \neq x'$ y $h(x)=h(x')$
- Si hay n+1 mensajes y n valores de salida, existe al menos una colisión

Propiedades (definición informal)

- Resistencia a preimagenes
  - Para todo *y*, es computacionalmente imposible hallar x / $h(x) = y$

Propiedades (definición informal)

- Resistencia a segundas imagenes
  - Para todo x, es computacionalmente imposible hallar x' ≠ x / h(x')=h(x)

Propiedades (definición informal)

- Resistencia a colisiones
  - Es computacionalmente imposible hallar x, x' / h(x) = h(x') y x ≠ x'

Resistencia a colisiones

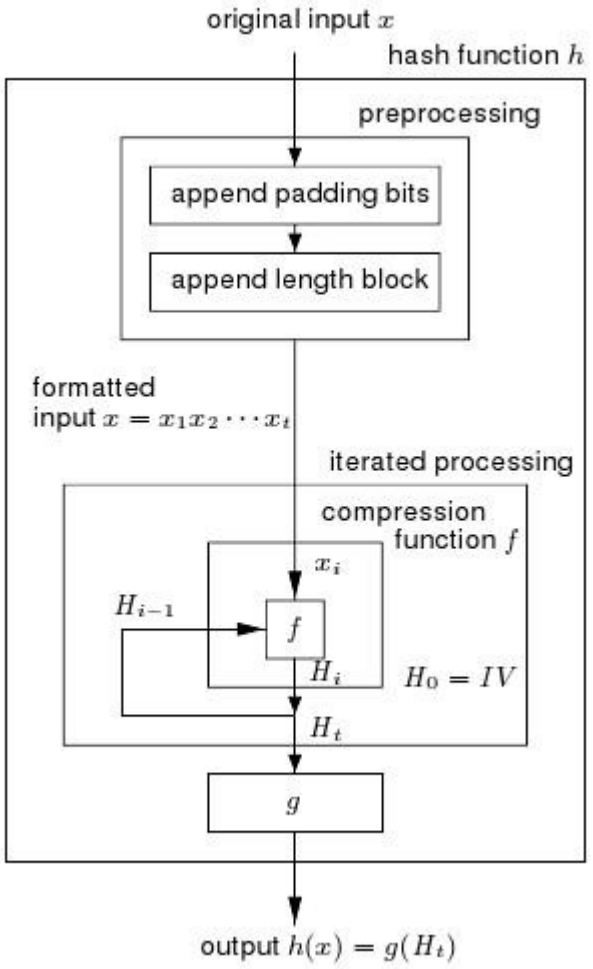
- Collision resistance: $\text{Hash-Coll}_{A,H}$
- Dado un nivel de seguridad n, un adversario A, y una familia de funciones de hash $H^x(n)$:

1) Se selecciona una funcíon $s \leftarrow S$
2) A obtiene s, y por lo tanto $H(x) = H^s(x)$
3) A emite x, x'

- $\text{Hash-Coll}_{A,H} = 1$ si $x \neq x'$ y $H^s(x) = H^s(x')$ (A encuentra una colisión)
- Si $\Pr[\text{Hash-Coll}_{A,H} = 1] < \text{neg}(n)$, la función H es libre de colisiones

Modelo general iterativo

- Propuesto por Merkle en 1989
- Utilizado por muchas funciones
- Preprocesamiento
  - Ajustar el tamaño del mensaje
  - Agregar bloque con tamaño
- Función de compresión
  - Similar al hash pero opera sobre bloques pequeños
  - La seguridad está dada por la función de compresión



MD5

- Entrada: secuencia de hasta 2^{64} bits
- Salida: secuencia de 128 bits
- Aplicación iterativa
- Ejemplos:
  - “” -> `d41d8cd98f00b204e9800998ecf8427e`
  - “a” -> `0cc175b9c0f1b6a831c399e269772661`
  - “abc” -> `900150983cd24fb0d6963f7d28e17f72`

SHA-1

- Entrada: secuencia de hasta 2^{64} bits
- Salida: secuencia de 160 bits
- Aplicación iterativa
- Se construye sobre la base de MD5
- Ejemplos:
  - “” -> `da39a3ee5e6b4b0d3255bfef95601890afd80709`
  - “a” -> `86f7e437faa5a7fce15d1ddcb9eaeaea377667b8`
  - “abc” -> `a9993e364706816aba3e25717850c26c9cd0d89d`

SHA-3

- Entrada: secuencia de hasta 2^{64} bits
- Salida: secuencia de 224,256,384 o 512 bits
- Estandarizado por NIST en 2013
- Aplicación modelo esponja
- Ejemplos:
  - “” -> `a7ffc6f8bf1ed76651c14756a061d662f580ff4de43b49fa82d80a4b80f8434a`
  - “a” -> `80084bf2fba02475726feb2cab2d8215eab14bc6bdd8bfb2c8151257032ecd8b`
  - “abc” → `3a985da74fe225b2045c172d6bd390bd855f086e3e9d525b46bfe24511431532`

Funciones de Hash y MACs

- Es posible construir un MAC a partir de una función de hash
- HMAC:
  - Dado H(x) función de hash libre de colisiones
  - HMACk(x) = {

    GEN: $k \leftarrow K, s \leftarrow S$

    MAC: $t = H^s((k \oplus opad) \| H^s((k \oplus ipad) \| m))$

    opad = 0x36...36 ipad= 0x5c5c...5c

    }

    Es infalsificable (MAC-Forge)

Seguridad de funciones de Hash

- Objetivos del atacante. Sea h: A → B
  - Preimagenes: dado $y \in B$, hallar x / $h(x) = y$
    - *Búsqueda por fuerza bruta: requiere* $|B|$ *intentos*
  - *Segundas imagenes: dado (x,y) / h(x)=y, hallar x' / h(x')=y*
    - *Búsqueda por fuerza bruta: requiere* $|B|$ *intentos*
  - *Colisiones: hallar x, x' / h(x)=h(x')*
    - *Búsqueda por fuerza bruta: requere* $|B|^{1/2}$
    - *Paradoja del cumpleaños*

Funciones de hash en la práctica

- Tamaño mínimo de salida: 160 bits
  - Complejidad de un ataque por fuerza bruta:
    - Imágenes: 2^{160} operaciones
    - Colisiones: 2^{80} operaciones
- Primitivas recomendadas
    - ~~MD5 → 128 bits~~ Quebrada
    - SHA-1 → 160bits
    - SHA-2/256 → 256 bits
    - SHA-3/256-512: Recientemente seleccionada
      - Antes conocida como Kekkak
      - Estandard recomendado para nuevos proyectos

Privacidad e integridad

- Tres formas:
  - Cifrar y autentificar:
    - $c \leftarrow Enc_{k1}(m), t \leftarrow Mac_{k2}(m)$

    t puede brindar información de m
  - Autentificar, luego cifrar
    - $c \leftarrow Enc_{k1}(m \,||\, Mac_{k2}(m))$,

    Puede ser seguro, requiere prueba de seguridad
  - Cifrar, luego Autentificar
    - $c \leftarrow Enc_{k1}(m), t \leftarrow Mac_{k2}(c)$

    Siempre es seguro

Cifrado Autentificado

- Combina cifrado y control de integridad
- Sean $\Pi_e(Gen_e, Enc, Dec)$ un criptosistema CPA-Secure y $\Pi_m(Gen_m, mac, vrfy)$ un mac infalsificable.
- Se define el criptosistema:
  - Gen: $k_1 \leftarrow Gen_e$, $k_2 \leftarrow Gen_m$
  - Enc: $c \leftarrow Enc_{k1}(m)$, $t \leftarrow Mac_{k2}(c)$
  - Dec: Si $vrfy_{k2}(c, t) = 1 \rightarrow m = Dec_{k1}(c)$

    Sino, m = / (fallo)
- El criptosistema resultante es CCA-Secure

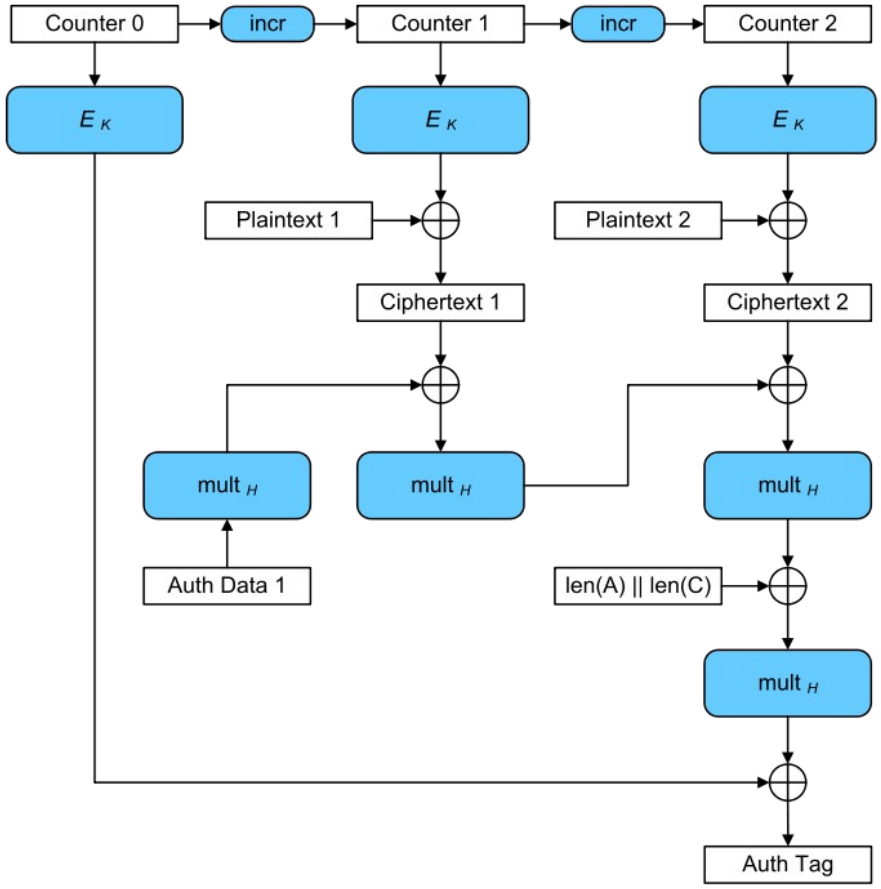
Encadenamiento CCM

- Significa Counter with CBC-MAC
  - “Authenticate-then-encrypt”
  - $e_k(\text{cbc-mac}_k(M) \,||\, m)$
- Existe una prueba de seguridad especifica
  - Si el IV y el nonce no coinciden ni se reutilizan, la construcción es CCA-Secure usando LA MISMA CLAVE

    (ver material adicional en Campus).
- Ejercicio, esquematizar un cifrado utilizando AES-CCM

Encadenamiento GCM

- Forma de encadenar un criptosistema de bloque
- Provee cifrado autenticado



Cifrado modo counter

Auth tag (MAC) GHASH: $GF(2^{128})$

$H = E_k(0000...0000)$

$Mult_h(x) = Mult(x, h)$

$Mult(x, y) = x * y \bmod x^{128}+x^7+x^2+x+1$

Aplicaciones prácticas

- En términos generales, solo utilizar cifrado autentificado en sistemas reales
  - El cifrado “normal” puede ser manipulado
  - En aplicaciones reales, el requerimiento de privacidad lleva implícito el de integridad
  - Muy pocas veces se tiene control sobre el material que va a ser cifrado y descifrado. La solución debería funcionar independientemente de ésto.

Lectura Recomendada

Capítulo 4

Introduction to Modern Cryptography
Katz & Lindell