Universidad de Oviedo

Departamento de Informática

Campus de Gijón

Autenticación de mensajes

Presentación

Daniel F. García

Introducción

La autenticación de mensajes **es** un procedimiento para verificar que un mensaje recibido proviene realmente del remitente indicado y que no ha sido modificado

(También se puede verificar la secuenciación y los plazos temporales)

Los mecanismos de autenticación de mensajes incluyen 2 aspectos:
- Algún tipo de función en el emisor que genera un autenticador
- Un protocolo en el receptor que usa el autenticador

Clases de funciones que pueden usarse para generar el autenticador:

1. Función Hash

   Función que mapea un mensaje de cualquier longitud en un resumen de longitud fija que sirve como autenticador

2. Cifrado del mensaje

   El texto cifrado de todo el mensaje sirve como su autenticador

3. Código de autenticación del mensaje

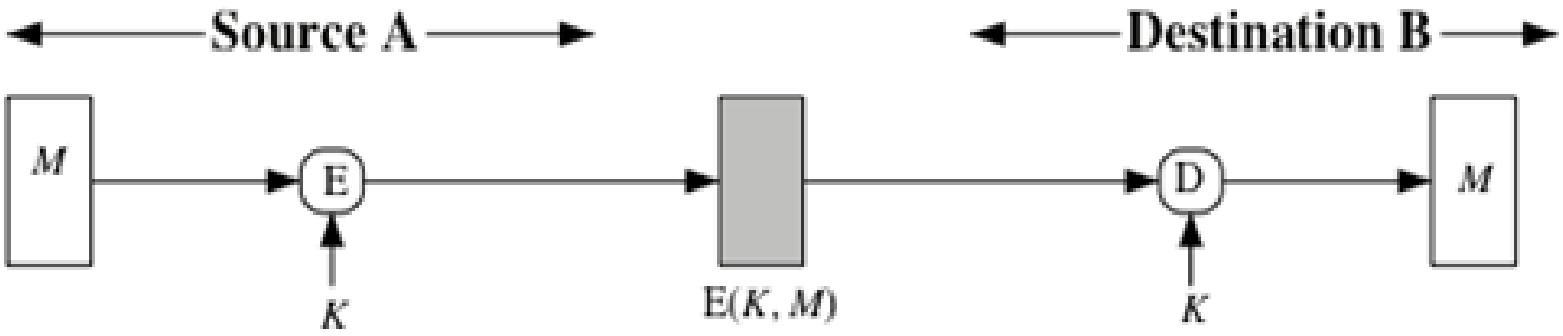
   Una función del mensaje junto con una clave secreta produce un valor de longitud fija que sirve como autenticador

Autenticación mediante el cifrado del mensaje (1)

El propio cifrado del mensaje puede proporcionar un mecanismo de autenticación

El análisis depende del tipo de cifrado: simétrico o de clave pública

Cifrado simétrico



(a) Symmetric encryption: confidentiality and authentication

Confidencialidad → M se transmite cifrado y se supone que solo A y B conocen la clave K

Autenticación (1) → B sabe que M ha sido generado por A, pues solo A tiene K (además de B)

Autenticación (2) → B sabe que M no ha sido alterado si la información recuperada es correcta

(2) Solo es aceptable si B puede determinar la corrección de la información recuperada

Ej. M contiene frases en lenguaje humano ó es un documento estructurado en XML

Autenticación mediante el cifrado del mensaje (2)

Si el M original es un patrón arbitrario de bits → el M recibido también es un patrón arbitrario

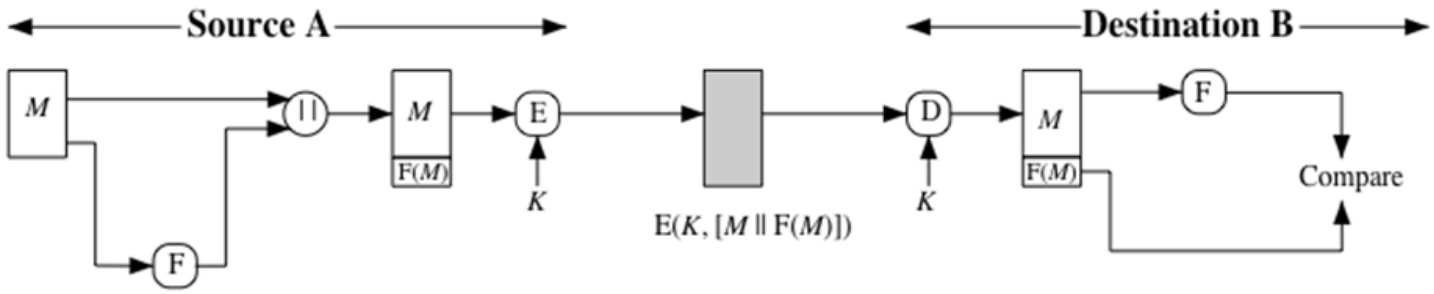
B no tiene posibilidad alguna de distinguir un mensaje alterado o de otro remitente

Solución:

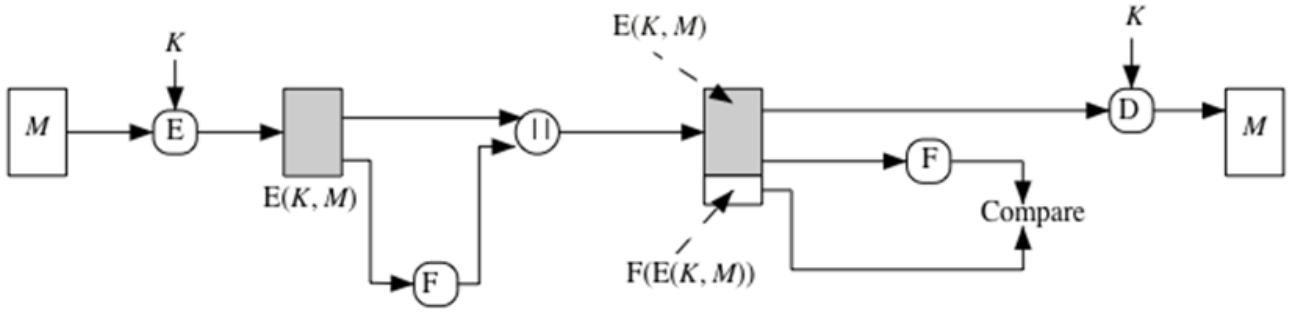
Añadir a M un código de detección de errores (*checksum*) → F(M)

a) Antes de cifrar

b) Después de cifrar



(a) Internal error control

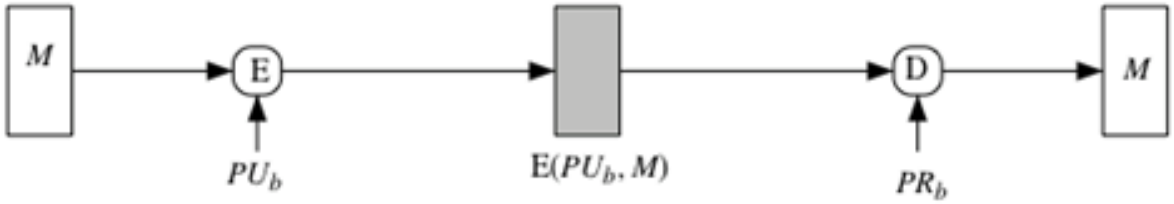


(b) External error control

Autenticación mediante el cifrado del mensaje (3)

Cifrado asimétrico (de clave pública)

Cifrar con la clave pública del destinatario B

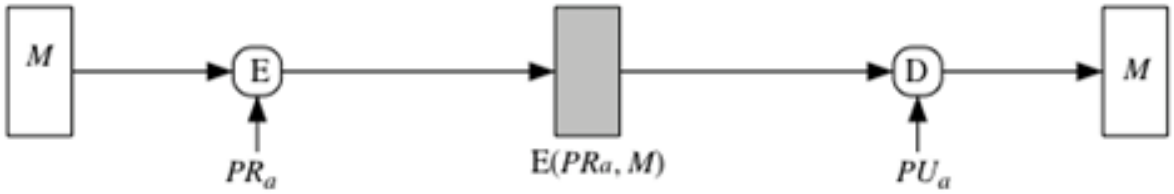


(b) Public-key encryption: confidentiality

Proporciona confidencialidad

NO Proporciona autenticación
Cifra cualquiera con PUb

Cifrar con la clave privada del emisor A



(c) Public-key encryption: authentication and signature

NO proporciona confidencialidad
Descifra cualquiera con PUa

Proporciona autenticación
Solo A conoce PRa

M integro si B lo ve correcto
(Leng. Humano o estructura)

Cifrar con PRa + Cifrar con PUb

(d) Public-key encryption: confidentiality, authentication, and signature

Lo proporciona todo
Pero es muy ineficiente

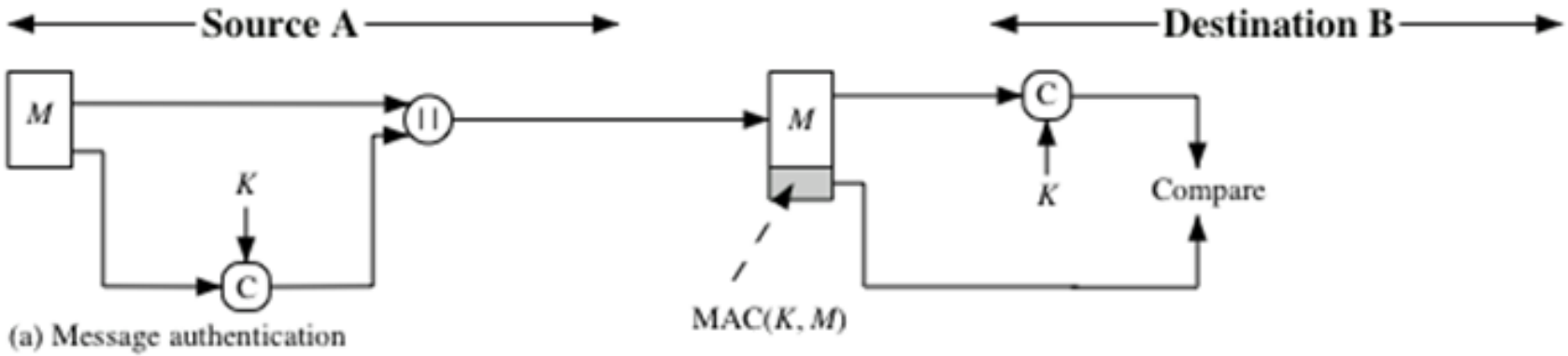
Autenticación mediante MAC (1)

MAC (*Message Authentication Code*) = Código de autenticación de mensajes

Un MAC es un pequeño bloque de información usado para autenticar un mensaje

El emisor y el receptor del mensaje deben compartir una clave secreta K

Mensaje M (cualquier tamaño) → Clave secreta K (tamaño fijo) → [Algoritmo MAC: MAC = C(K,M)] → Código MAC



A calcula el MAC de M, lo concatena con M (||) y lo envía todo a B

B calcula el MAC del M recibido y lo compara con el MAC recibido → SI coinciden

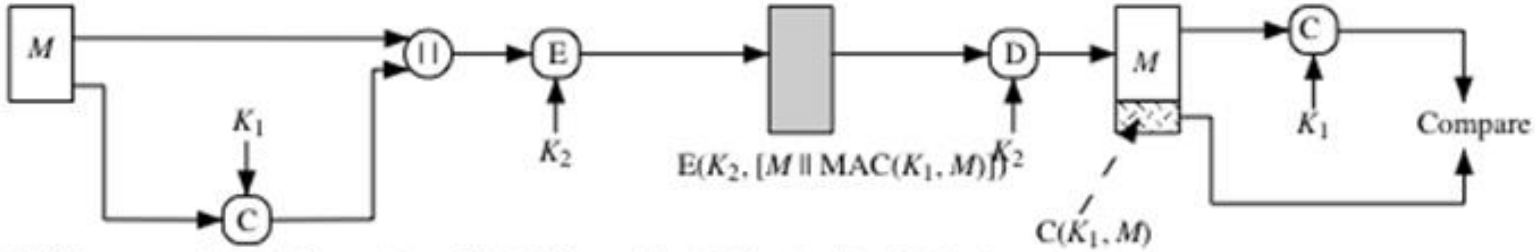
1.- B asume que M no fue alterado (solo A tiene K para recalcular el MAC)

2.- B asume que A es el remitente de M (solo A tiene K para generar el MAC recibido)

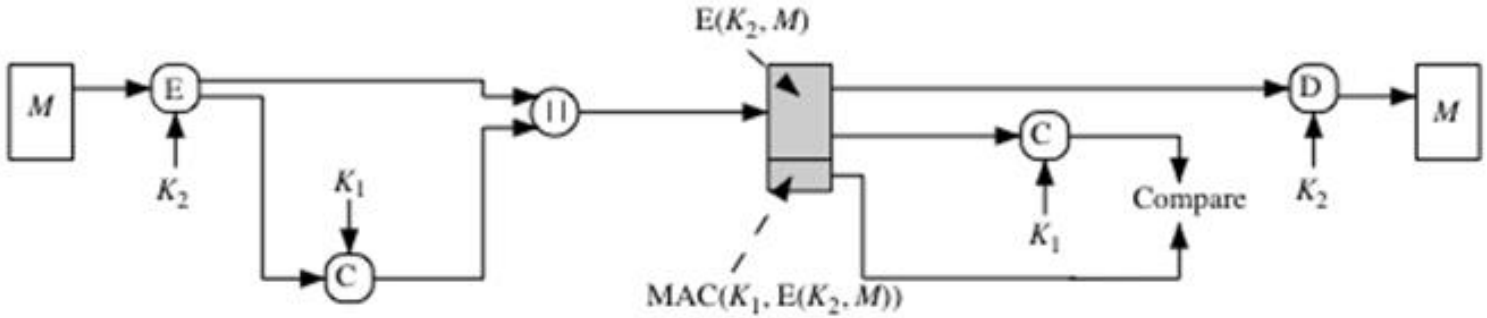
Si M incluye un número de secuencia, se puede comprobar si faltan mensajes

Autenticación mediante MAC (2)

El uso de MAC solo proporciona autenticación → Para obtener confidencialidad hay que cifrar



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Un MAC no proporciona una genuina firma digital (solo el que tiene K puede comprobar el origen)

Función MAC ← → Cifrador - Pero el Algoritmo MAC no necesita ser reversible (no se descifra)

Tipos de Funciones MAC → { Basadas en una función Hash ; Basadas en un algoritmo de cifrado simétrico }

HMAC – MAC basado en funciones Hash (1)

Las funciones HMAC se construyen usando funciones Hash

HMAC == Hash-Based MAC

Las funciones de Hash vistas …

- No han sido diseñadas para autenticación pues carecen de clave secreta
- Pero son muy útiles porque:
  - Su velocidad es mayor que la de los cifradores en bloque
  - Su código fuente es abierto
  - Sus propiedades y forma de operar son bien conocidas

Aunque ha habido múltiples propuestas para integrar una clave en una función de Hash …

La función HMAC utilizada actualmente fue propuesta en la RFC 2104 en Feb1997

https://www.rfc-editor.org/rfc/rfc2104

El NIST la adoptó como estándar para la Administración Federal (USA) en Jul2008

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf

HMAC – MAC basado en funciones Hash (2)

Las funciones HMAC utilizan una clave K de longitud apropiada

La clave K es compartida por el emisor y el receptor del mensaje a autenticar con HMAC

La función de Hash que usa HMAC procesa bloques de b bits (B Bytes)

F. Hash	Tamaño del Bloque	Tamaño del Hash
MD5	512 bits – 64 Bytes	128 bits – 16 Bytes
SHA 1	512 bits – 64 Bytes	160 bits – 20 Bytes
SHA 224	512 bits – 64 Bytes	224 bits – 28 Bytes
SHA 256	512 bits – 64 Bytes	256 bits – 32 Bytes
SHA 384	1024 bits – 128 Bytes	384 bits – 48 Bytes
SHA 512	1024 bits – 128 Bytes	512 bits – 64 Bytes

Internamente, la F. HMAC usa una clave K^+ obtenida adaptando la clave K en función del tamaño de bloque ***B*** que procesa la F. de Hash:

SI (Longitud K == ***B***) $K^+ = K$

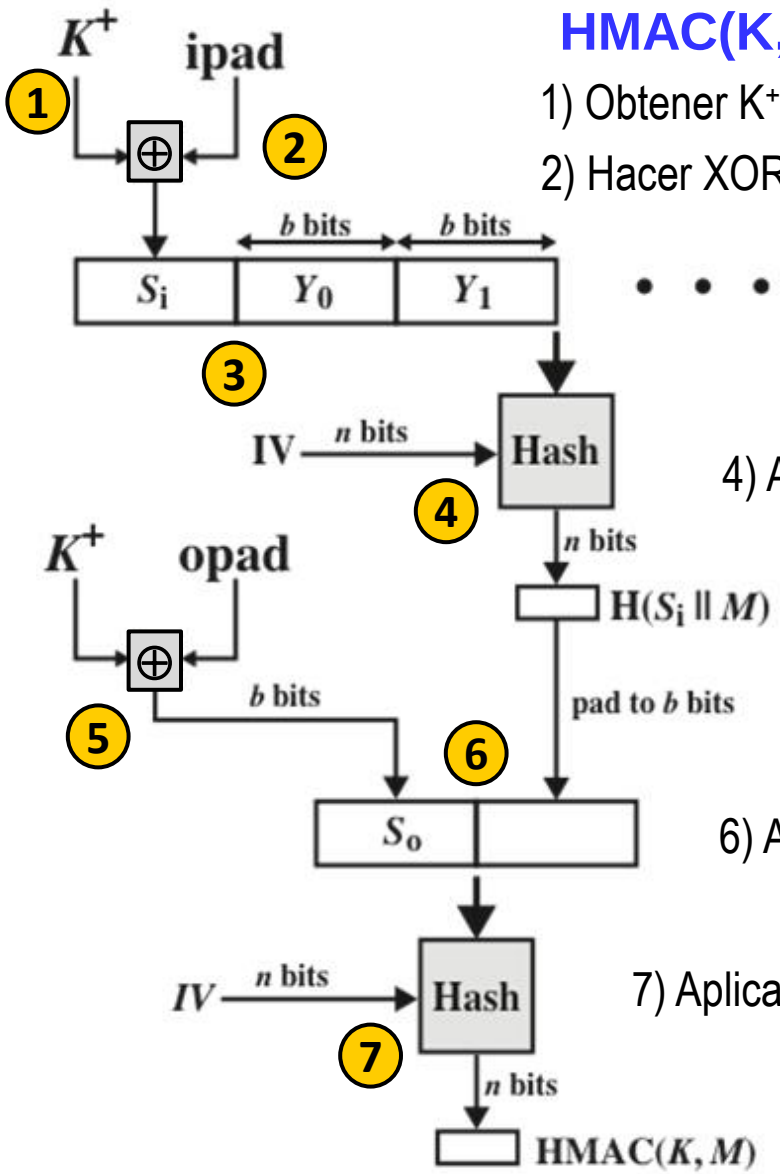
SI (Longitud K > ***B***) $K^+ = 00 \ldots 00 \ || \ Hash(K)$

Si se obtiene un hash de longitud L bytes se añaden B-L bytes cero por la izquierda

SI (Longitud K < ***B***) $K^+ = 00 \ldots 00 \ || \ K$

Si añaden ceros directamente por la izquierda

HMAC – MAC basado en funciones Hash (3)



HMAC(K,M) = H [(K⁺ ⊕ opad) || H((K⁺ ⊕ ipad) || M)]

1) Obtener K^+ de b bits a partir de K

2) Hacer XOR de K^+ con ipad → Bloque S_i de b bits

3) Añadir el mensaje M ($Y_0 \cdots Y_{L-1}$) a S_i

4) Aplicar la función de Hash a la cadena generada en el paso 3

5) Hacer XOR de K^+ con opad → Bloque S_o de b bits

6) Añadir el hash del paso 4 a S_o

7) Aplicar la función de Hash a la cadena generada en el paso 6

HMAC – MAC basado en funciones Hash (4)

Objetivo de los rellenos

ipad = 0011 0110 (36h) repetido ***B*** veces

opad = 0101 1100 (5Ch) repetido ***B*** veces

Invierten la mitad de los bits de K^+ → Proporcionan aleatoriedad adicional

Generan un bloque pseudoaleatorio (S_i y S_o) basado en la clave K para iniciar la cadena de bloques de entrada a cada función de Hash

Seguridad de HMAC

Depende de la seguridad del algoritmo de Hash que utiliza

Se ha podido probar una relación directa entre la seguridad de la función HMAC y la función Hash que utiliza

CMAC – MAC basado en Cifrador (1)

Las funciones CMAC se construyen usando cifradores de bloque

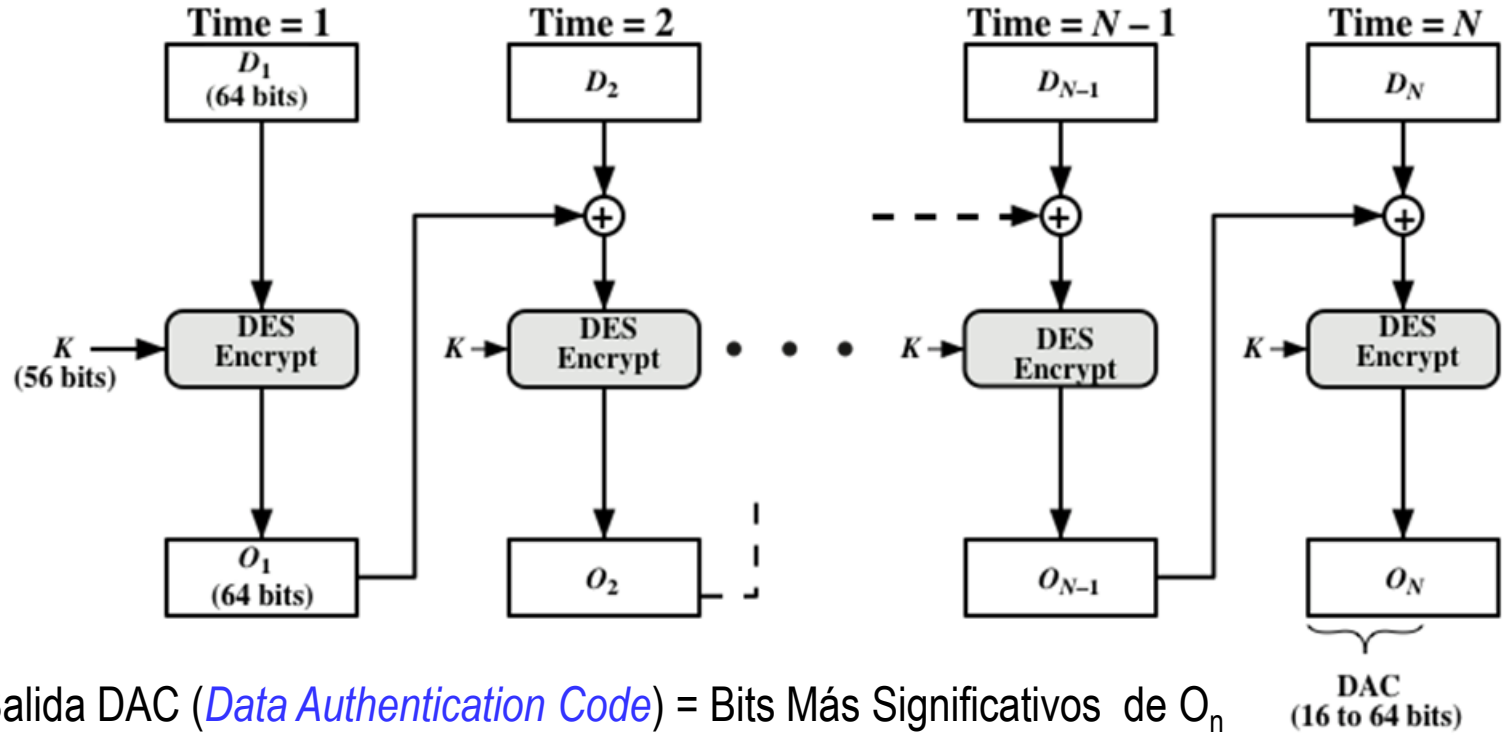
CMAC == Cipher-Based MAC

Data Authentication Algorihm (DAA)

Ha sido una función MAC usada durante muchos años

Es el estándar FIPS-113 y ANSI X9.17

$O_1 = E(K, D_1)$
$O_2 = E(K, [D_2 \oplus O_1])$
$O_3 = E(K, [D_3 \oplus O_2])$
…
$O_n = E(K, [D_n \oplus O_{n-1}])$



Salida DAC (*Data Authentication Code*) = Bits Más Significativos de O_n

CMAC – MAC basado en Cifrador (2)

MAC basado un cifrador de bloques (CMAC)

Es una función MAC especificada en NIST SP-800-38B

Utiliza los cifradores de bloque 3DES ó AES

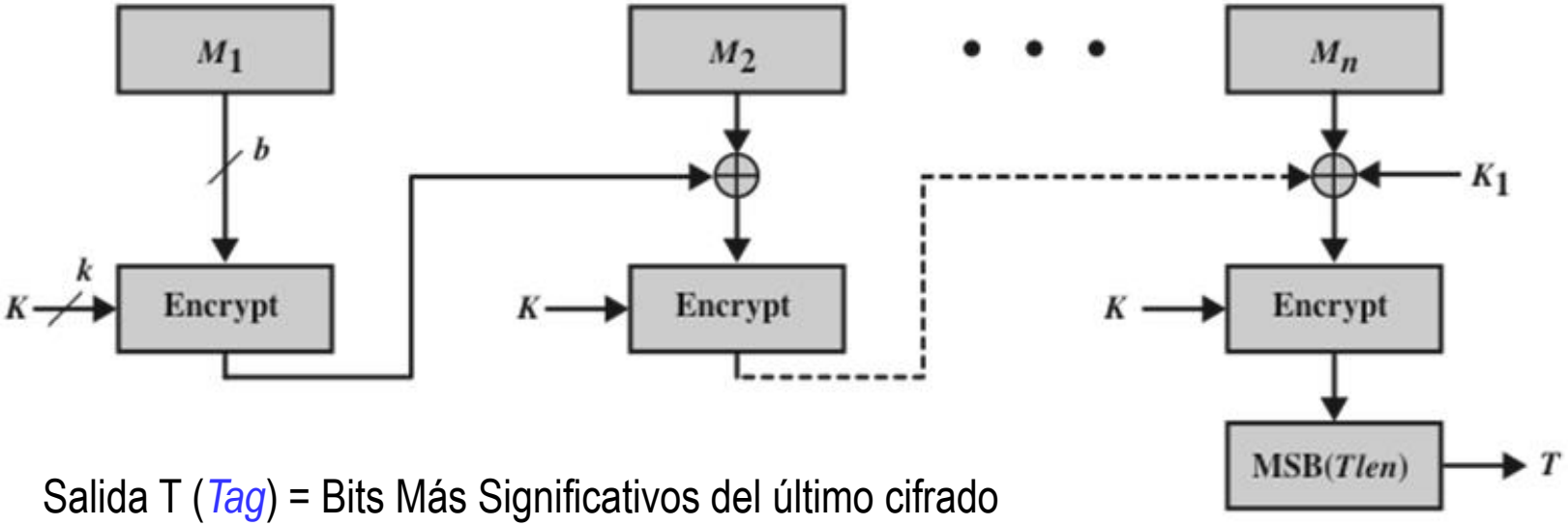
Para AES → b=128 k=128, 192, 256

Para 3DES → b=64 k=112, 168

CMAC ≈ DAA pero se aleatoriza el último bloque con K_1

$$C_1 = E(K, M_1)$$
$$C_2 = E(K, [M_2 \oplus C_1])$$
$$C_3 = E(K, [M_3 \oplus C_2])$$
$$\ldots$$
$$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$$
$$T = MSB_{LT}(C_n)$$

Si el tamaño del mensaje es un entero múltiplo del tamaño del bloque del cifrador (b bits)



Salida T (*Tag*) = Bits Más Significativos del último cifrado

CMAC – MAC basado en Cifrador (3)

Si el tamaño del mensaje NO es un entero múltiplo del tamaño del bloque del cifrador (b bits)

1) Se rellena el último bloque del mensaje
2) Para el último bloque se usa K_2 en vez de K_1

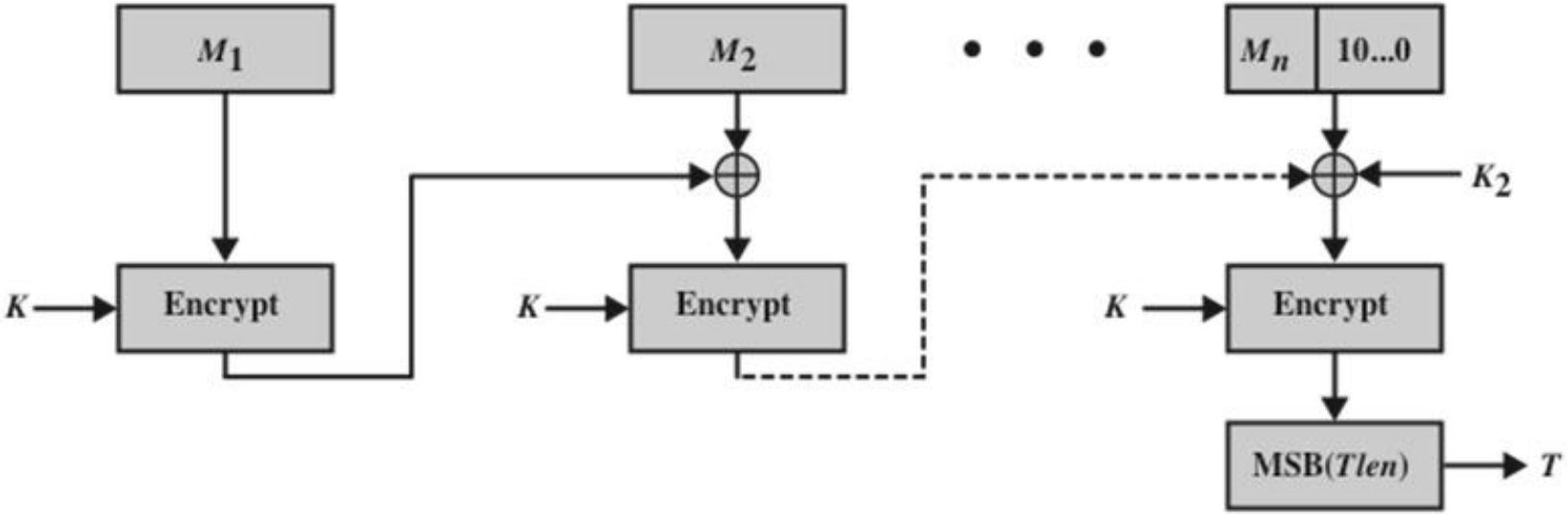
Derivación de las claves K_1 y K_2 de la clave K →

$$\begin{cases} L = E(K, 0) \\ K_1 = L \cdot x \\ K_2 = L \cdot x^2 = (L \cdot x) \cdot x \end{cases}$$

Se cifra con la clave K un bloque de ceros

L = Valor cifrado interpretado como polinomio en $GF(2^b)$

· = Producto de polinomios en $GF(2^b)$



GCM – Galois/Counter Mode (1)

GCM es un algoritmo para el cifrado autenticado de datos (*Authenticated Encryption*)

Su variante GMAC genera un código MAC sin cifrar los datos

El NIST define GCM como un modo de operación para cifradores de bloques

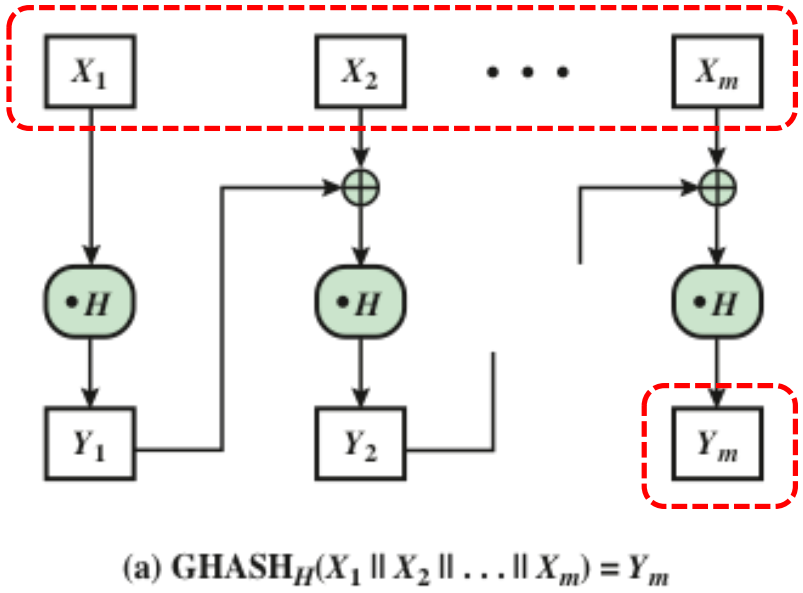
La especificación del modo GCM la publicó el NIST en noviembre de 2007:
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf

GCM se basa en dos funciones:

- GHASH que es una función hash con clave (~HMAC)
- GCTR que usa un cifrador de bloque simétrico en el modo Counter (CTR)

GCM – Galois/Counter Mode (2)

Función GHASH



(a) $\text{GHASH}_H(X_1 \| X_2 \| \ldots \| X_m) = Y_m$

Entrada: Cadena de bits
Organizada en bloques Xi de 128 bits

Salida: bloque (hash) de 128 bits

Utiliza una clave H en todas las etapas

En cada etapa i se multiplica la entrada (Xi $\oplus$ Yi-1) por H
(multiplicación realizada en GF(2^{128}) – GF = Galois Field)

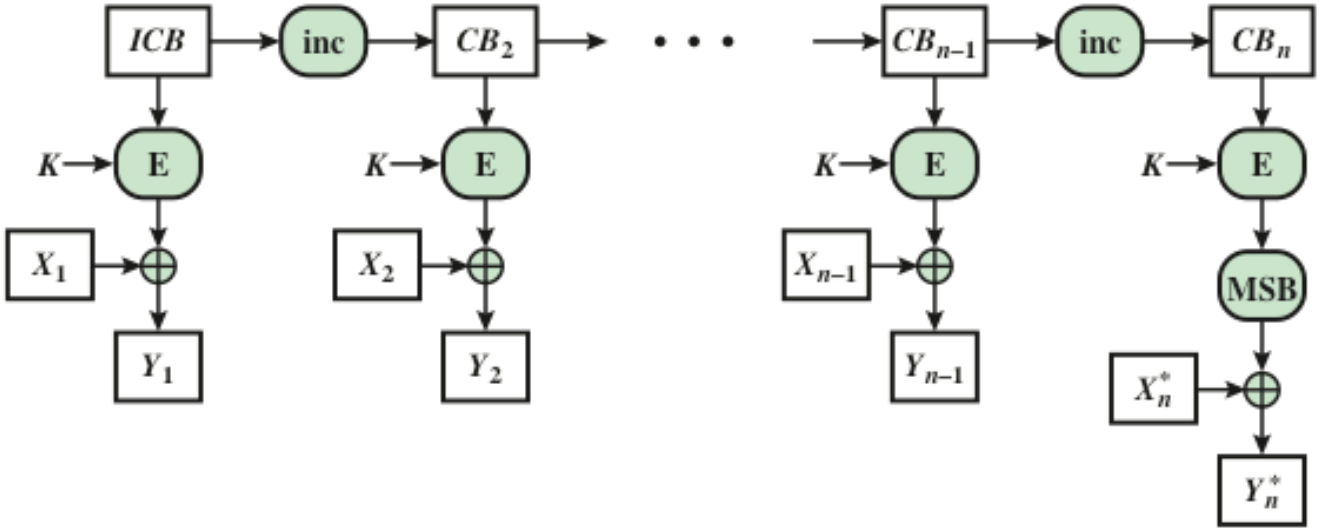
Operación eficiente:

$\text{GHASH}_H(X) = (X_1 \cdot H^m) \oplus (X_2 \cdot H^{m-1}) \oplus \cdots \oplus (X_{m-1} \cdot H^2) \oplus (X_m \cdot H)$

Precalcular H, $H^2 \cdots H^m$ → Hacer todas las operaciones en paralelo

GCM – Galois/Counter Mode (3)

Función GCTR



(b) $\text{GCTR}_K(ICB, X_1 \| X_2 \| \ldots \| X_n) = Y_1 \| Y_2 \| \ldots \| Y_n$

Los bloques cifradores (E) generan una secuencia de cifrado a partir de ICB

La entrada a los cifradores E_2 E_3 … se obtiene incrementando el contador previo

CBi = inc_{32}(CBi-1) inc_{32} solo incrementa los 32 bits + signif de CBi-1

Los bloques cifrados Yi = Xi [bloque plano] XOR E(K, CBi) [secuencia de cifrado]

GCM – Galois/Counter Mode (4)