UNIVERSIDAD INTERNACIONAL SEK
ECUADOR

SEGURIDAD DE DATOS

Ing. José Luis Medina

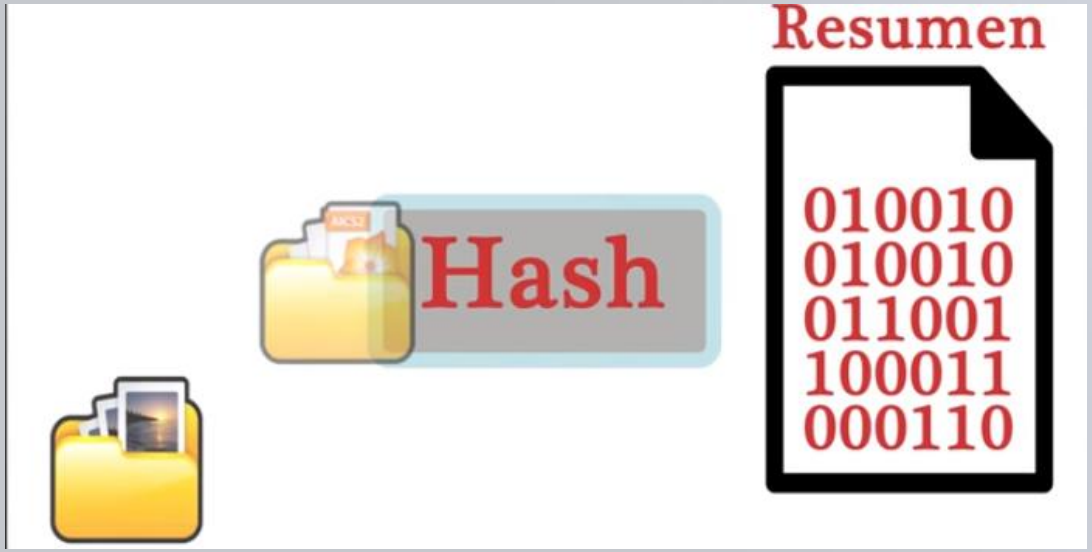
Integridad de Datos

Integridad de Datos

- Significa que un mensaje que atravieza un **canal público**, no sea modificado en su trayecto de llegada al destino
- Para ello se puede hacer el uso de funciones hash para comprobar que el mensaje **no haya sido modificado**
- Entre las funciones hash que importantes tenemos:
- MD5
- SHA
- SHA – 1, SHA – 2, SHA - 3

- Una función hash es un algoritmo que aplicado a un texto o archivo devuelve como resultado un resumen de **X** bits, este, es un número que puede ser visto como una huella digital del archivo supuestamente única
- A diferencia de los programas zip, la función hash entrega siempre un resumen de **X** bits independientemente del tamaño de dicho archivo

- Una función hash **NO** es un algoritmo de cifrado



Message Digest 5 (MD5)

- Es un algoritmo de resumen desarrollado en 1991 por Ron Rivest , fue un algoritmo de los más usados por su alta sencillez y su alta velocidad
- Hoy en día es un algoritmo ya **obsoleto** desde mediados de 2005

- El algoritmo **MD5** entrega un resumen de **128 bits** independientemente del tamaño del mensaje o archivo

Algoritmo básico de Message Digest 5

a) Un mensaje M se convierte en un bloque múltiplo de 512 bits, añadiendo bits si es necesario al final del mismo.

b) Con los 128 bits de cuatro vectores iniciales ABCD de 32 bits cada uno y el primer bloque del mensaje de 512 bits, se realizan diversas operaciones lógicas entre ambos bloques.

c) La salida de esta operación (128 bits) se convierte en el nuevo conjunto de 4 vectores A'B'C'D' y se realiza la misma función con el segundo bloque de 512 bits del mensaje, y así hasta el último bloque del mensaje. Al terminar, el algoritmo entrega un resumen que corresponde a los últimos 128 bits de estas operaciones.

http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html

Etapas de MD5

Bloques funcionales de MD5

a) Añadir bits para congruencia módulo 512, reservando los últimos 64 bits para un indicador de longitud.

b) Añadir indicación de la longitud del mensaje en los 64 bits reservados para ello.

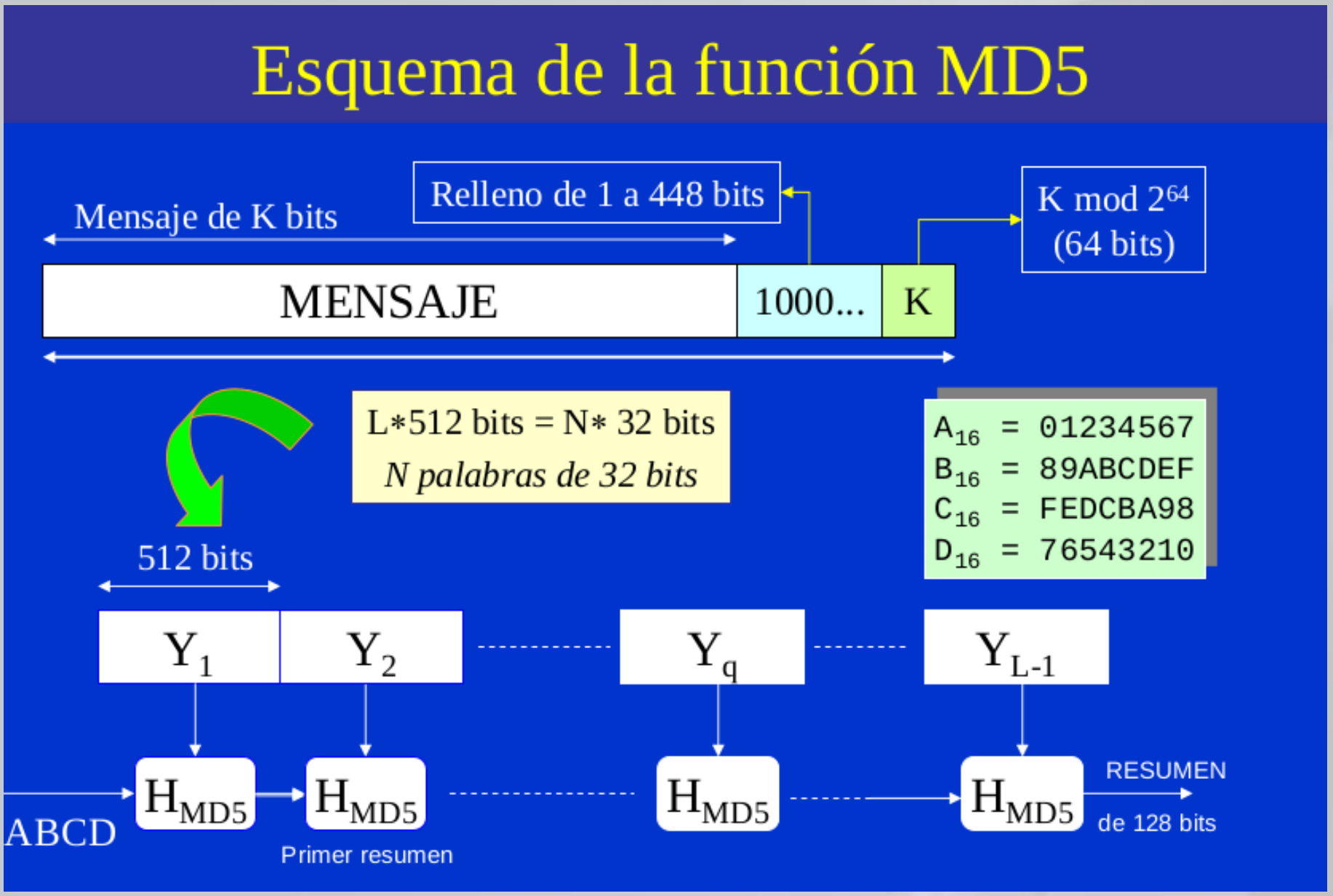
c) Inicializar el vector ABCD de claves con un valor que no es secreto.

d) Procesar bloques de 512 bits, entregando una salida de 128 bits que formarán nuevamente el vector ABCD.

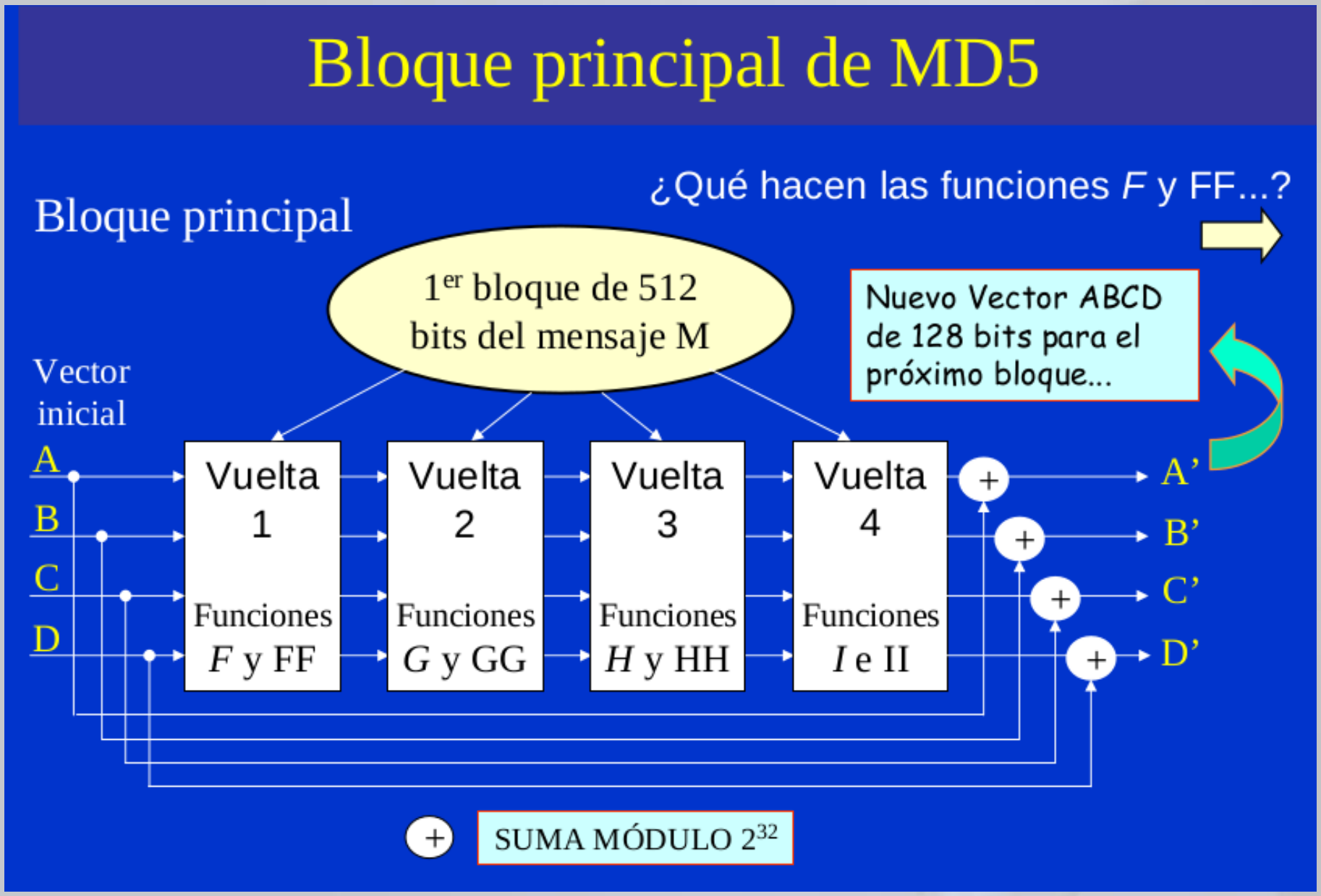
e) Obtener el resumen de los últimos 128 bits.

http://www.faqs.org/rfcs/rfc1321.html

Esquema de la función MD5



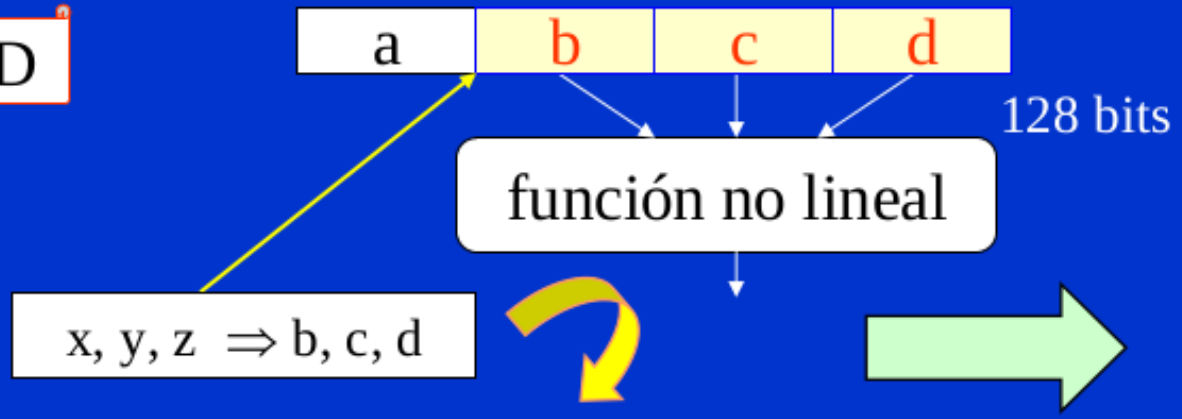
Bloque principal de MD5



Esquema funciones F, G, H, I en MD5

Vector inicial ABCD

A_{16} = 01234567
B_{16} = 89ABCDEF
C_{16} = FEDCBA98
D_{16} = 76543210



F (x, y, z)
(x AND y) OR (NOT x AND z)
G (x, y, z)
(x AND z) OR (y AND NOT z)
H (x, y, z)
x XOR y XOR z
I (x, y, z)
y XOR (x OR NOT z)

F (b, c, d)
(b AND c) OR (NOT b AND d)
G (b, c, d)
(b AND d) OR (c AND NOT d)
H (b, c, d)
b XOR c XOR d
I (b, c, d)
c XOR (b OR NOT d)

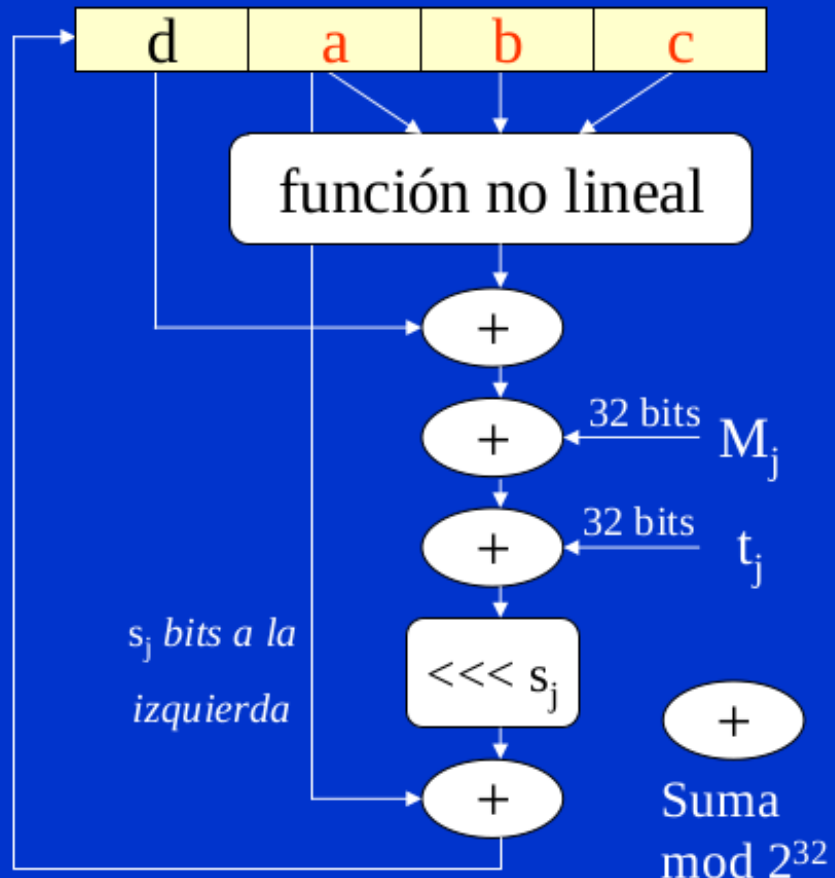
Algoritmo de las funciones en MD5

Desplazamiento del registro

Situación luego del desplazamiento

Se repite el proceso para M_{j+1} hasta 16 bloques del texto. En las vueltas 2, 3 y 4 se repite el proceso ahora con funciones G, H e I.

El algoritmo realiza 4∗16 = 64 vueltas para cada uno de los bloques de 512 bits



Funciones no lineales en MD5

Funciones no lineales en cada vuelta

Vector de 128 bits

a	b	c	d

1ª Vuelta:

$FF(a,b,c,d,M_j,t_j,s) \Rightarrow a = b + ((a + F(b,c,d) + M_j + t_j) <<< s)$

2ª Vuelta:

$GG(a,b,c,d,M_j,t_j,s) \Rightarrow a = b + ((a + G(b,c,d) + M_j + t_j) <<< s)$

3ª Vuelta:

$HH(a,b,c,d,M_j,t_j,s) \Rightarrow a = b + ((a + H(b,c,d) + M_j + t_j) <<< s)$

4ª Vuelta:

$II(a,b,c,d,M_j,t_j,s) \Rightarrow a = b + ((a + I(b,c,d) + M_j + t_j) <<< s)$

```
joseluis@themordor:~$ echo "MAESTRIA EN CIBERSEGURIDAD" | md5sum
42bf601d5e9d47df300066f93fcc664e  -
joseluis@themordor:~$
```

```
joseluis@themordor:~$ md5sum mensaje.txt.gpg
d89f4ea640b0b1116de583a773c7fc98  mensaje.txt.gpg
joseluis@themordor:~$
```

Codificar un MD5

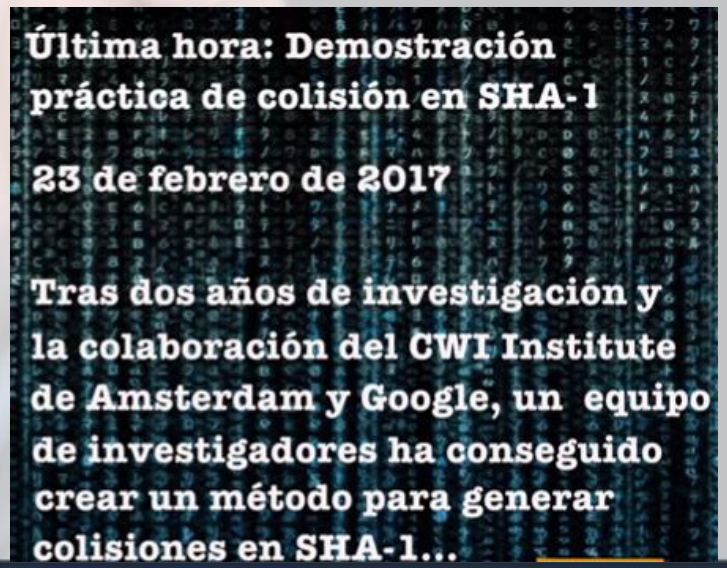
Escribe una palabra aquí para obtener su hash MD5 :

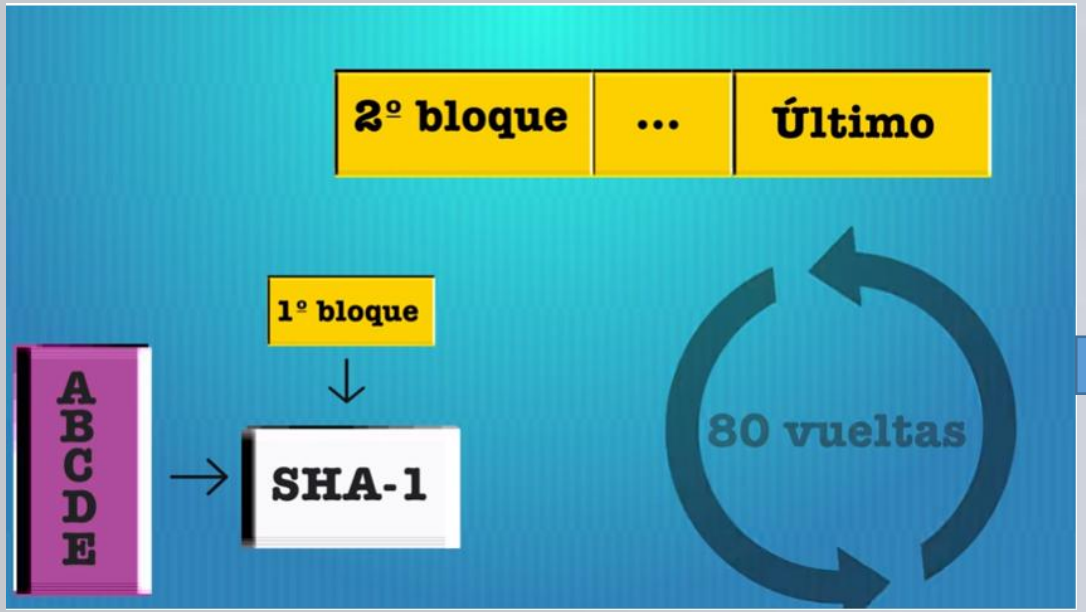
Cifrar

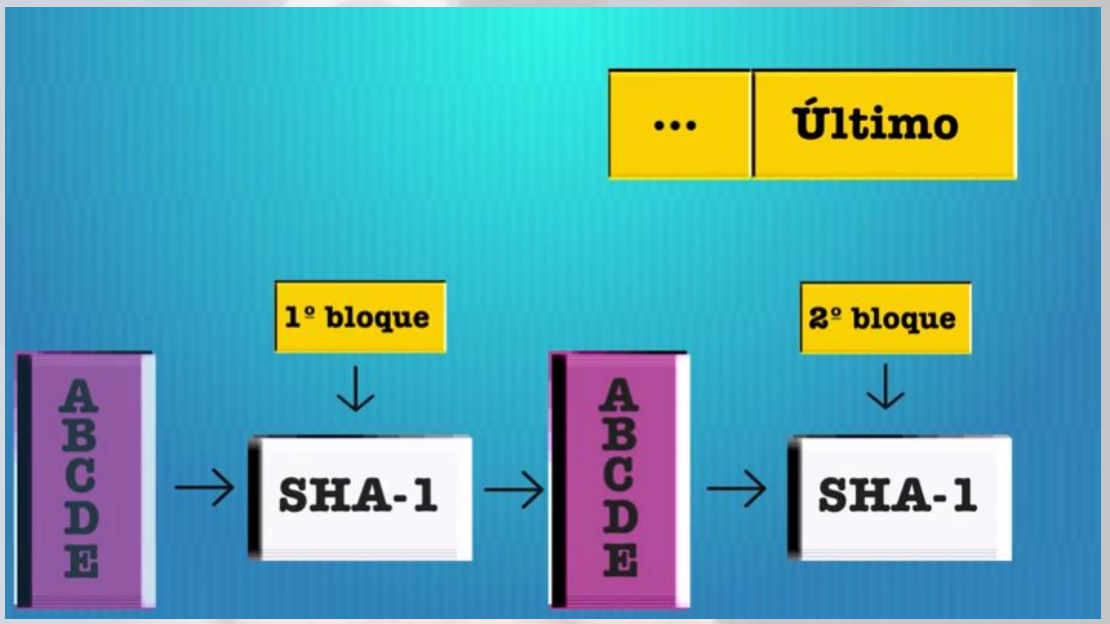
El hash MD5 correspondiente a MAESTRIA EN CIBERSEGURIDAD es : **6fce4477f9a25090e3711434ea3ebbfa**

Secure Hash Algorithm - SHA

- SHA es un algoritmo de hash seguro, es un algoritmo propuesto inicialmente por la NSA en 1993 y que posteriormente fue adoptado por la NIST
- En 1995 se descubren vulnerabilidades al algoritmo por lo que se propone un nuevo algoritmo mejorado SHA1
- SHA1 es un algoritmo similar a MD5, pero este entregará un resumen de 160 bits







Registro de 160 bits

A_{16} = 67452301
B_{16} = EFCDAB89
C_{16} = 98BADCFE
D_{16} = 10325476
E_{16} = C3D2E1F0

Secure Hash Algorithm – SHA 2

- En 2001 la **NSA** genera una serie de algoritmos **HASH** conocidos como **SHA2** que consiste en una serie de funciones sha-224, sha-256, sha-384 y sha-512 bits
- Este estándar sigue **vigente** puesto que **NO** se ha encontrado vulnerabilidades, sin embargo, la **NIST** ya tiene un nuevo estándar vigente también que es **SHA-3**

Algoritmo y variante		Tamaño del hash (bits)	Longitud de la palabra (bits)	Tamaño del bloque (bits)	Vueltas	Tamaño del vector interno (bits)	Tamaño máximo del mensaje (bits)
SHA-2	*SHA-224*	224	32	512	64	256 (8 x 32)	$2^{64} - 1$
	SHA-256	256					
	SHA-384	384	64	1.024	80	512 (8 x 64)	$2^{128} - 1$
	SHA-512	512					
	SHA-512/224	224					
	SHA-512/256	256					

```
joseluis@themordor:~$ sha1sum mensaje.txt.gpg
304dca2416bf97ca997b5cef99eb3e516f19cc94  mensaje.txt.gpg
joseluis@themordor:~$
```

```
joseluis@themordor:~$ echo "MAESTRIA EN CIBERSEGURIDAD" | sha1sum
d512435764d3884ed072a4237c20a799c6c114e0  -
joseluis@themordor:~$
```

```
joseluis@themordor:~$ echo "MAESTRIA EN CIBERSEGURIDAD" | sha256sum
3d4a283d898e7f7288bcaac7b8ae35479fe22d17301173eabd6ce4122b25a776  -
joseluis@themordor:~$
```

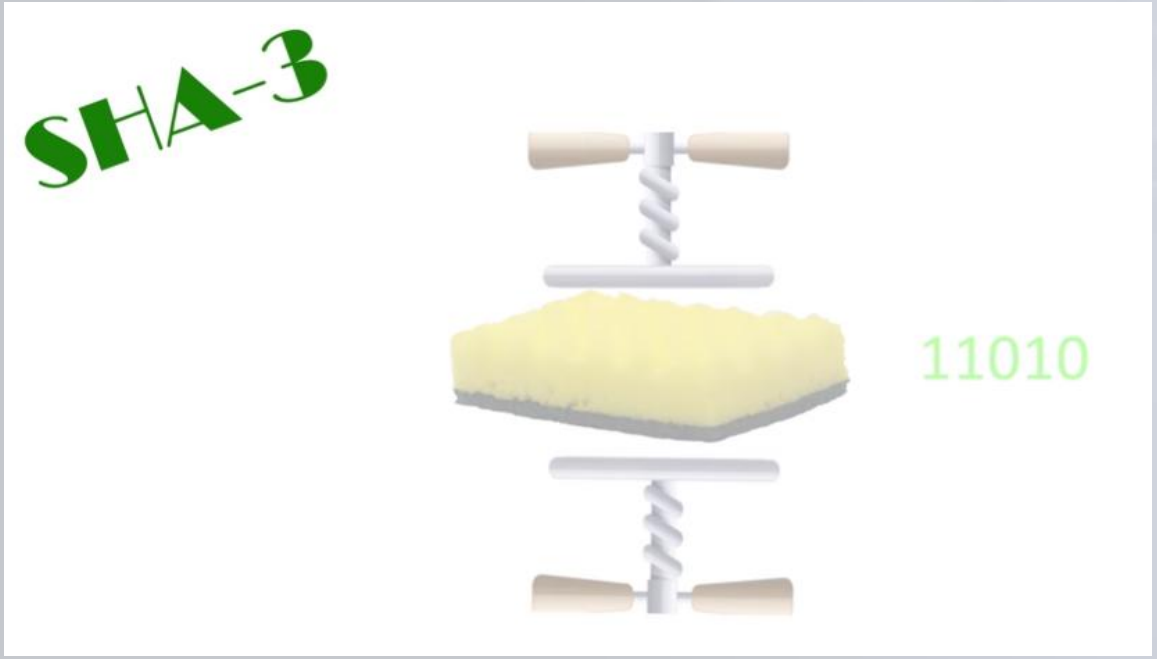
```
joseluis@themordor:~$ sha256sum  mensaje.txt.gpg
d7a2a9a0122ada6af39c9291910b192ee93764a8ff5fe0c59a873d086fd55d1c  mensaje.txt.gpg
joseluis@themordor:~$
```

Secure Hash Algorithm – SHA 3

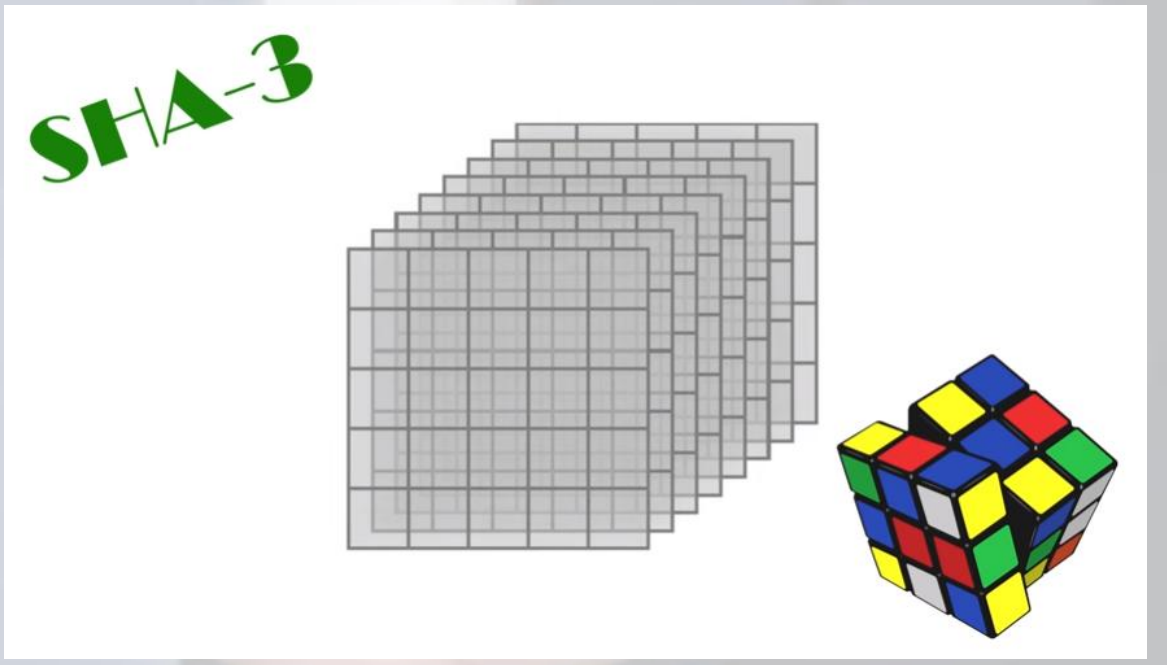
- Se basa en un nuevo algoritmo llamado KECCAK, el mismo que se basa en funciones ESPONJA con lo cual se aleja radicalmente de la familia de las funciones MD4/5 y de SHA-2
- Keccak fue diseñado por un equipo de criptógrafos de Bélgica e Italia, estos son:
- Guido Bertoni (Italia) de STMicroelectronics,
- Joan Daemen (Bélgica) de STMicroelectronics,
- Michaël Peeters (Bélgica) de NXP Semiconductors, y
- Gilles Van Assche élgica) de STMicroelectronics.

- Esta función **ESPONJA** en una permutación de longitud fija (o transformación) y una regla de relleno, que construye una función de mapeo de longitud de entrada variable a salida de longitud variable. Toma como entrada un elemento de $(Z_2)^*$, es decir una cadena binaria de cualquier longitud, en otras palabras $(Z_2)^n$ con un N valor proporcionado por el usuario
- Una función esponja es una generalización d efunciones HASH, que tiene una longitud de salida fija y cifrados de flujo, que tienen una longitud fija de entrada

Algoritmo **ESPONJA**



Cálculo




```
joseluis@themordor:~$ echo "MAESTRIA EN CIBERSEGURIDAD" | rhash --sha3-256 -
799cab19d6714c54444220ca36928657cce1a49525069420dba2b593d6ed602f  (stdin)
joseluis@themordor:~$ echo "MAESTRIA EN CIBERSEGURIDAD" | rhash --sha3-512 -
0fe852cd8b50aaeb9eeae4f549a8e293b1a0ea9fecb881f1292e61fac3484640859adce780f673945beff083a230d76b1e47d6171a00934ffcd85ef021c53818  (stdin)
joseluis@themordor:~$
```

Bibliografía

- STALLINGS WILLIAM, Cryptographic and Network Security, Principles and Practice, Five Edition

- AGUIRRE RAMIO, JORGE, Libro Electrónico de Seguridad Informática y Criptografía , Sexta Edición (SLIDES, tomadas de este libro en la parte MD5)